MERROWDALE UNIVERSITY

The Graduate School · Office of Graduate Admissions

Application for Graduate Admission

Fall 2027 entry

APPLICANT

Name	**Aiko Whitcombe**	Email	**aiko.whitcombe@mail.example**
Applicant ID	**M31902052**	Telephone	**(555) 0143-8676**
Date of birth	**07 Jul 1997**		
Citizenship	**Kingdom of Estmark**		
Mailing address	**892 Quarry Street, Tarnbrook, Ohio**		

PROGRAM

Degree sought	**Ph.D. in Computer Science**
Department	**Department of Computer Science**
Area of interest	**Distributed consensus**
Entry term	**Fall 2027**
Enrollment	**Full-time**
Financial support	**Self-funded**
Faculty of interest	**Prof. Whitcombe and Prof. Achterberg**

ACADEMIC HISTORY

Institution	Degree or status	Attended	GPA
Royal Estmark University	B.S. in Computer Science	2018-2022	3.38
Tarnbrook College	M.S. in Computer Science	2022-2024	3.40

TEST SCORES

Test	Section	Score	Percentile	Date
Graduate Aptitude Examination	Verbal Reasoning	780	95	26 Mar 2026
Graduate Aptitude Examination	Quantitative Reasoning	670	71	26 Mar 2026
Graduate Aptitude Examination	Analytical Writing	5.0	91	26 Mar 2026
Test of Academic English	Listening	25		16 Apr 2026
Test of Academic English	Reading	29		16 Apr 2026
Test of Academic English	Writing	27		16 Apr 2026
Test of Academic English	Speaking	28		16 Apr 2026

RECOMMENDERS

Name	Position	Institution	Status
Prof. Naomi Fairweather	Associate Professor	Royal Estmark University	Received
Prof. Viktor Hrabal	Professor	Tarnbrook College	Received
Prof. Helena Marchetti	Assistant Professor	Royal Estmark University	Received

UPLOADED DOCUMENTS

Document type	File name	Uploaded
Research Proposal	proposal_draft4.pdf	05 Nov 2026
Financial Documentation	scan_2969.pdf	08 Nov 2026
Resume / CV	Whitcombe_Aiko_CV_2026.pdf	11 Nov 2026
Transcript	scan_9690.pdf	25 Oct 2026
Transcript	scan_4976.pdf	03 Nov 2026
Passport / ID	passport_scan.pdf	23 Nov 2026
English Proficiency Scores	english_test.pdf	01 Nov 2026
Writing Sample	Whitcombe_sample.pdf	19 Nov 2026
Statement of Purpose	Whitcombe_SOP.pdf	24 Oct 2026
Test Scores	scores.pdf	17 Nov 2026
Degree Certificate	diploma.pdf	04 Nov 2026
Letter of Recommendation	letter_284.pdf	03 Nov 2026
Letter of Recommendation	LOR_Hrabal.pdf	21 Nov 2026
Letter of Recommendation	letter_433.pdf	15 Nov 2026

SUPPLEMENTAL QUESTIONS

How do you intend to fund your studies?

After graduating I spent two years in industry, where questions about query optimisation came up almost every week. In the longer term I hope to lead applied research on program synthesis, either in a public agency or at a university.

SUPPLEMENTAL QUESTIONS (CONTINUED)

Have you previously applied to this university?

Reading the recent literature on program synthesis, I was struck by how many basic questions remain open. My coursework gave me a broad foundation, but it left little time to follow one question in depth.

Describe any gaps in your academic record.

In particular, I hope to study how fault tolerance interacts with distributed consensus under realistic conditions. My interest in cache behaviour began during my second year of undergraduate study.

DISCLOSURES

Previously enrolled at this university	**No**
Academic or disciplinary action	**None reported**
Application fee	**Paid online**

CERTIFICATION

I certify that the information given in this application is complete and accurate, and that the documents uploaded with it are true copies of the originals. I understand that admission may be withdrawn if any part of this application is found to be false.

Electronically signed: Aiko Whitcombe 30 Nov 2026

Research Proposal

Static analysis reconsidered

Aiko Whitcombe · Ph.D. in Computer Science · Merrowdale University

1. Aims

It would be a mistake, however, to read this result as evidence against the importance of cache behaviour. Several objections can be raised at this point, and the most serious concerns the treatment of program synthesis. That explanation cannot account for the size of the change, which is far larger than any plausible error.

Aim 1. To establish how cache behaviour varies across sites and over time.

Aim 2. To establish how fault tolerance varies across sites and over time.

Aim 3. To establish how distributed consensus varies across sites and over time.

2. Background

The second case is harder to interpret, because static analysis changed at the same time. What remains unexplained is why fault tolerance responded so slowly. Even so, the overall direction of the results is not in doubt. Taken together, these observations point to a more gradual process than the standard account allows. Three cases illustrate the argument, and each was chosen because its records are unusually complete. The estimates are sensitive to the choice of baseline, a point taken up again below.

The limits of the evidence should be stated plainly. The contrast with cache behaviour is instructive. The data were collected over three seasons and cross-checked against published records. It is tempting to generalise from these examples, and the temptation should be resisted.

None of this implies that distributed consensus is unimportant; it implies that its effect depends on context. Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. Comparable patterns appear in neighbouring regions, although the timing differs by several years. One explanation is that query optimisation was simply measured more carefully in the later period. The relationship between query optimisation and fault tolerance has rarely been examined at this scale. This reading is consistent with what practitioners have reported informally for some time.

In practical terms, the finding suggests that work on query optimisation should begin earlier than is now usual. A simple comparison of the two periods shows a clear shift in distributed consensus. A second line of argument holds that distributed consensus is better understood as a consequence of cache behaviour than as its cause. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. The approach adopted in this paper is deliberately conservative.

This distinction matters because measures aimed at distributed consensus have often produced unintended effects on static analysis. The remainder of this section sets out the assumptions on which the analysis rests. The next section turns from description to explanation. Further work on fault tolerance would need a longer series than the one available here.

Earlier studies relied on small samples, which limits what can be said about fault tolerance in general. Three cases illustrate the argument, and each was chosen because its records are unusually complete. The next section turns from description to explanation. The data were collected over three seasons and cross-checked against published records. Several objections can be raised at this point, and the most serious concerns the treatment of query optimisation. Comparable patterns appear in neighbouring regions, although the timing differs by several years.

What remains unexplained is why cache behaviour responded so slowly. The remainder of this section sets out the assumptions on which the analysis rests. The contrast with fault tolerance is instructive. Even so, the overall direction of the results is not in doubt. None of this implies that cache behaviour is unimportant; it implies that its effect depends on context. The approach adopted in this paper is deliberately conservative.

3. Approach

The limits of the evidence should be stated plainly. A fuller account would also have to consider query optimisation, which lies outside the scope of this paper. Taken together, these observations point to a more gradual process than the standard account allows. Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise.

It is tempting to generalise from these examples, and the temptation should be resisted. Where the two measures diverge, the difference can usually be traced to how query optimisation was recorded. One explanation is that distributed consensus was simply measured more carefully in the later period. In practical terms, the finding suggests that work on static analysis should begin earlier than is now usual.

A second line of argument holds that query optimisation is better understood as a consequence of cache behaviour than as its cause. That explanation cannot account for the size of the change, which is far larger than any plausible error. It would be a mistake, however, to read this result as evidence against the importance of query optimisation. The relationship between query optimisation and distributed consensus has rarely been examined at this scale.

A simple comparison of the two periods shows a clear shift in program synthesis. Earlier studies relied on small samples, which limits what can be said about cache behaviour in general. This distinction matters because measures aimed at distributed consensus have often produced unintended effects on cache behaviour. The estimates are sensitive to the choice of baseline, a point taken up again below.

The second case is harder to interpret, because static analysis changed at the same time. Further work on program synthesis would need a longer series than the one available here. This reading is consistent with what practitioners have reported informally for some time. The contrast with cache behaviour is instructive.

Taken together, these observations point to a more gradual process than the standard account allows. The second case is harder to interpret, because program synthesis changed at the same time. Further work on fault tolerance would need a longer series than the one available here. In practical terms, the finding suggests that work on static analysis should begin earlier than is now usual.

The limits of the evidence should be stated plainly. Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. Three cases illustrate the argument, and each was chosen because its records are unusually complete. Even so, the overall direction of the results is not in doubt. A second line of argument holds that program synthesis is better understood as a consequence of distributed consensus than as its cause.

The relationship between query optimisation and fault tolerance has rarely been examined at this scale. Earlier studies relied on small samples, which limits what can be said about distributed consensus in general. This reading is consistent with what practitioners have reported informally for some time. One explanation is that static analysis was simply measured more carefully in the later period.

It is tempting to generalise from these examples, and the temptation should be resisted. A simple comparison of the two periods shows a clear shift in fault tolerance. The data were collected over three seasons and cross-checked against published records. The next section turns from description to explanation. None of this implies that program synthesis is unimportant; it implies that its effect depends on context.

The approach adopted in this paper is deliberately conservative. That explanation cannot account for the size of the change, which is far larger than any plausible error. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. The remainder of this section sets out the assumptions on which the analysis rests. This distinction matters because measures aimed at program synthesis have often produced unintended effects on fault tolerance.

The estimates are sensitive to the choice of baseline, a point taken up again below. Comparable patterns appear in neighbouring regions, although the timing differs by several years. What remains unexplained is why program synthesis responded so slowly. Where the two measures diverge, the difference can usually be traced to how static analysis was recorded. Several objections can be raised at this point, and the most serious concerns the treatment of fault tolerance.

It would be a mistake, however, to read this result as evidence against the importance of query optimisation. Taken

together, these observations point to a more gradual process than the standard account allows. Earlier studies relied on small samples, which limits what can be said about fault tolerance in general. The remainder of this section sets out the assumptions on which the analysis rests.

The second case is harder to interpret, because cache behaviour changed at the same time. It is tempting to generalise from these examples, and the temptation should be resisted. Where the two measures diverge, the difference can usually be traced to how cache behaviour was recorded. A simple comparison of the two periods shows a clear shift in cache behaviour. Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise. The relationship between distributed consensus and fault tolerance has rarely been examined at this scale.

4. Timeline

Stage	When	Duration
Literature review and site selection	Year 1	6 months
Pilot study	Year 1	9 months
Main data collection	Year 2	5 months
Analysis	Year 2	7 months
Writing and defence	Year 3	8 months

5. References

Delacroix, K. (2003). What fault tolerance can tell us about distributed consensus. Studies in Method and Evidence, 9(3), 11-33.

Delacroix, L. (2025). Measuring fault tolerance: a case study. Journal of Applied Inquiry, 19(4), 206-223.

Mendoza, H. (2007). What static analysis can tell us about cache behaviour. Quarterly Review of Field Studies, 51(2), 39-55.

Sorensen, R., Okonkwo, B., Lindgren, R. (2015). A long view of cache behaviour and static analysis. Annals of Regional Research, 47(3), 94-104.

MERROWDALE UNIVERSITY

AFFIDAVIT OF FINANCIAL SUPPORT

For international applicants. Complete in ink and attach a recent bank statement.

Applicant name

Aiko Whitcombe

Applicant ID

M31902052

Program

Ph.D. in Computer Science

Sponsor name

Daniel Whitcombe

Relationship to applicant

Parent

Sponsor address

Brennhaven, Kingdom of Estmark

Amount pledged for each academic year (USD)

43,500

Number of years of support

2

I declare that I am willing and able to provide the support stated above for the full period of study, and that the attached bank documents are genuine and show funds available to me.

Daniel Whitcombe

Signature of sponsor, 15 Nov 2026

NOTARY PUBLIC

HARBOURGATE SAVINGS BANK

Member of the Harbourgate Group · harbourgate.example

Statement of Account

Account holder	**Daniel Whitcombe**	Opening balance	**248,527.00**
Account number	****** **** 2904**	Closing balance	**258,875.78**
Period	**12 Aug 2026 to 12 Dec 2026**	Currency	**EST**

Date	Description	Withdrawals	Deposits	Balance
12 Aug	Insurance premium	477.71		248,049.29
14 Aug	Rent payment	992.51		247,056.79
14 Aug	Transfer to savings	604.66		246,452.12
14 Aug	Insurance premium	529.05		245,923.07
14 Aug	Salary credit		5,148.58	251,071.65
15 Aug	Utility payment	703.56		250,368.09
16 Aug	Utility payment	685.61		249,682.48
17 Aug	Restaurant	1,025.20		248,657.29
18 Aug	Utility payment	842.89		247,814.39
19 Aug	Rent payment	1,365.00		246,449.40
20 Aug	Cash withdrawal	979.57		245,469.83
22 Aug	Rent payment	1,086.21		244,383.62
25 Aug	Fuel	811.26		243,572.36
25 Aug	Utility payment	994.29		242,578.07
28 Aug	Utility payment	908.02		241,670.05
30 Aug	Transfer received		1,632.32	243,302.37
31 Aug	Fuel	1,063.80		242,238.57
01 Sep	Utility payment	1,344.54		240,894.03
02 Sep	Cash withdrawal	250.65		240,643.38
03 Sep	Rail fare	35.66		240,607.72
05 Sep	Transfer received		2,473.24	243,080.97
08 Sep	Salary credit		2,424.33	245,505.30
11 Sep	Transfer received		904.37	246,409.67
11 Sep	Utility payment	44.79		246,364.88
14 Sep	Card payment, grocery	706.37		245,658.51
14 Sep	Rail fare	573.16		245,085.35
14 Sep	Card payment, grocery	360.06		244,725.29
15 Sep	Salary credit		7,303.33	252,028.61
16 Sep	Mobile top-up	743.64		251,284.97
18 Sep	Insurance premium	120.16		251,164.81

Daniel Whitcombe **Account **** **** 2904** **(continued)**

Date	Description	Withdrawals	Deposits	Balance
19 Sep	Utility payment	684.80		250,480.01
20 Sep	Transfer received		6,904.79	257,384.80
20 Sep	Cash withdrawal	154.87		257,229.93
23 Sep	Pharmacy	1,068.44		256,161.49
26 Sep	Insurance premium	798.83		255,362.67
26 Sep	Card payment, grocery	196.78		255,165.89
29 Sep	Transfer to savings	858.69		254,307.20
29 Sep	Fuel	1,333.86		252,973.34
29 Sep	Insurance premium	214.47		252,758.86
01 Oct	Online purchase	1,158.25		251,600.61
04 Oct	Rail fare	339.04		251,261.57
04 Oct	Cash withdrawal	843.70		250,417.87
05 Oct	Utility payment	684.56		249,733.32
08 Oct	Online purchase	768.03		248,965.29
09 Oct	Transfer received		4,856.51	253,821.79
09 Oct	Fuel	951.96		252,869.83
09 Oct	Salary credit		5,301.49	258,171.32
10 Oct	Interest credit		4,877.13	263,048.45
10 Oct	Rail fare	1,352.30		261,696.16
12 Oct	Card payment, grocery	913.22		260,782.94
14 Oct	Rent payment	751.62		260,031.32
14 Oct	Insurance premium	1,316.58		258,714.74
14 Oct	Transfer to savings	802.37		257,912.37
16 Oct	Mobile top-up	1,226.31		256,686.06
17 Oct	Transfer to savings	867.37		255,818.69
18 Oct	Insurance premium	1,047.50		254,771.19
18 Oct	Mobile top-up	1,051.55		253,719.64
21 Oct	Insurance premium	560.14		253,159.50
21 Oct	Transfer to savings	170.80		252,988.70
22 Oct	Transfer to savings	220.74		252,767.96
22 Oct	Mobile top-up	660.64		252,107.32
23 Oct	Card payment, grocery	769.35		251,337.97
24 Oct	Rail fare	599.29		250,738.68
24 Oct	Insurance premium	871.56		249,867.13
26 Oct	Online purchase	154.36		249,712.77
26 Oct	Rent payment	510.80		249,201.97

Daniel Whitcombe **Account **** **** 2904** **(continued)**

Date	Description	Withdrawals	Deposits	Balance
27 Oct	Transfer to savings	401.11		248,800.87
28 Oct	Restaurant	1,334.14		247,466.72
29 Oct	Utility payment	925.57		246,541.16
01 Nov	Mobile top-up	57.40		246,483.75
01 Nov	Fuel	1,027.82		245,455.93
04 Nov	Restaurant	763.33		244,692.59
06 Nov	Insurance premium	611.57		244,081.03
08 Nov	Rent payment	894.23		243,186.80
08 Nov	Utility payment	998.84		242,187.96
10 Nov	Card payment, grocery	1,175.29		241,012.67
13 Nov	Salary credit		967.75	241,980.42
14 Nov	Pharmacy	1,260.02		240,720.41
17 Nov	Rail fare	154.96		240,565.45
17 Nov	Online purchase	379.24		240,186.21
19 Nov	Rent payment	549.11		239,637.10
22 Nov	Pharmacy	259.93		239,377.16
22 Nov	Online purchase	1,006.78		238,370.38
22 Nov	Pharmacy	169.02		238,201.35
23 Nov	Insurance premium	154.01		238,047.35
23 Nov	Interest credit		8,953.65	247,001.00
24 Nov	Cash withdrawal	641.44		246,359.55
27 Nov	Card payment, grocery	503.12		245,856.43
27 Nov	Card payment, grocery	1,269.35		244,587.08
30 Nov	Fuel	1,057.65		243,529.43
02 Dec	Restaurant	1,096.37		242,433.06
02 Dec	Mobile top-up	354.12		242,078.94
02 Dec	Restaurant	470.06		241,608.87
03 Dec	Transfer to savings	797.54		240,811.33
05 Dec	Online purchase	332.76		240,478.58
05 Dec	Interest credit		5,773.50	246,252.07
08 Dec	Interest credit		8,607.33	254,859.41
08 Dec	Salary credit		2,971.03	257,830.44
09 Dec	Fuel	70.07		257,760.37
10 Dec	Utility payment	1,211.23		256,549.14
10 Dec	Salary credit		2,340.67	258,889.81
12 Dec	Mobile top-up	14.04		258,875.78

Closing balance 258,875.78 EST

Aiko Whitcombe

aiko.whitcombe@mail.example | (555) 0143-8676 | 892 Quarry Street, Tarnbrook, Ohio

EDUCATION

M.S. in Computer Science 2022-2024
Tarnbrook College, Tarnbrook, Ohio

B.S. in Computer Science 2018-2022
Royal Estmark University, Brennhaven, Kingdom of Estmark

RESEARCH EXPERIENCE

Teaching Assistant 2017-2019
Royal Estmark University, Department of Computer Science

- Organised a monthly reading group on program synthesis for students and staff.
- Reviewed more than eighty published studies of distributed consensus for a systematic summary.
- Designed and ran a twelve-week study of fault tolerance, collecting and cleaning the full data set.

Undergraduate Researcher 2020-2021
Tarnbrook College, Department of Computer Science

- Led weekly problem sessions for a class of forty students.
- Wrote the data management plan adopted by the whole group.
- Prepared figures and tables for a manuscript now under review.
- Maintained laboratory equipment and the group's shared documentation.

Laboratory Technician 2025-2026
Royal Estmark University, Department of Computer Science

- Co-wrote a technical report on program synthesis that was circulated to partner agencies.
- Presented interim findings on distributed consensus at two regional meetings.
- Trained four new assistants in sampling and record-keeping procedures.
- Supervised a team of three interns through a summer project on query optimisation.

Project Analyst 2023-2025
Tarnbrook College, Department of Computer Science

- Built analysis scripts that cut the time needed to process field data from days to hours.
- Presented interim findings on static analysis at two regional meetings.
- Trained four new assistants in sampling and record-keeping procedures.

PROFESSIONAL EXPERIENCE

Laboratory Technician 2023-present
Norvane Consulting Group, Portland, Maine

- Led weekly problem sessions for a class of forty students.
- Built analysis scripts that cut the time needed to process field data from days to hours.
- Maintained laboratory equipment and the group's shared documentation.

Laboratory Technician 2023-2024
Norvane Consulting Group, Portland, Maine

- Reviewed more than eighty published studies of program synthesis for a systematic summary.
- Wrote the data management plan adopted by the whole group.

PUBLICATIONS AND PRESENTATIONS

Whitcombe, A., Achterberg, B. (2024). Fault tolerance reconsidered. Review of Comparative Practice, 38(2), 129-162.

Sorensen, R., Whitcombe, A. (2024). Static analysis and the limits of fault tolerance. Studies in Method and Evidence, 49(2), 30-55.

Petrov, S., Whitcombe, A. (2022). What distributed consensus can tell us about fault tolerance. Annals of Regional Research, 34(2), 285-316.

Whitcombe, A., Reyes, B. (2023). Distributed consensus reconsidered. Quarterly Review of Field Studies, 40(2), 139-160.

TEACHING

Teaching Assistant Spring 2019

Database Systems, Royal Estmark University

- Designed and ran a twelve-week study of fault tolerance, collecting and cleaning the full data set.

Teaching Assistant Fall 2020

Machine Learning, Tarnbrook College

- Supervised a team of three interns through a summer project on query optimisation.

HONORS AND AWARDS

Honors College Graduate, 2019

Outstanding Teaching Assistant Award, 2017

Summer Research Stipend, 2023

Dean's List, six semesters, 2023

Departmental Prize for Best Senior Thesis, 2021

SKILLS

Statistical modelling and data visualisation

Experimental design and survey methods

Geographic information systems

ROYAL ESTMARK UNIVERSITY

Office of the Registrar · Bremnhaven, Kingdom of Estmark

OFFICIAL ACADEMIC TRANSCRIPT

Student	**Aiko Whitcombe**	Program	**B.S. in Computer Science**
Student ID	**426813**	Admitted	**Fall 2018**
Date of birth	**07 Jul 1997**	Issued	**12 Oct 2026**

Fall 2018

Course	Title	Credits	Grade	Points
PSYC 450	Introductory Psychology	3.0	A	12.0
COMP 166	Advanced Algorithms	3.0	B-	8.1
COMP 296	Advanced Software Engineering	3.0	A-	11.1
COMP 490	Seminar in Machine Learning	3.0	C	6.0
COMP 321	Advanced Theory of Computation	4.0	A-	14.8
		Term credits 16.0	Term GPA 3.25	Cumulative GPA 3.25

Spring 2019

Course	Title	Credits	Grade	Points
COMP 112	Software Engineering	3.0	A	12.0
SOCI 376	Social Theory	3.0	B+	9.9
COMP 399	Seminar in Theory of Computation	3.0	A	12.0
COMP 398	Compilers	4.0	A-	14.8
STAT 324	Introduction to Statistics	3.0	B+	9.9
		Term credits 16.0	Term GPA 3.66	Cumulative GPA 3.46

Fall 2019

Course	Title	Credits	Grade	Points
COMP 230	Data Structures	3.0	A-	11.1
COMP 248	Seminar in Compilers	3.0	B+	9.9
COMP 385	Computer Networks	3.0	C+	6.9
COMP 316	Advanced Database Systems	3.0	C+	6.9
COMP 341	Advanced Machine Learning	4.0	A	16.0
		Term credits 16.0	Term GPA 3.17	Cumulative GPA 3.36

Spring 2020

Course	Title	Credits	Grade	Points
ENGL 383	Academic Writing	3.0	B	9.0
COMP 218	Database Systems	3.0	B+	9.9
COMP 130	Seminar in Operating Systems	3.0	C	6.0
LANG 347	Intermediate Spanish	3.0	A-	11.1
PHIL 121	Ethics	4.0	A-	14.8
		Term credits 16.0	Term GPA 3.17	Cumulative GPA 3.32

Official only when it bears the signature of the Registrar and the seal of the institution.

OFFICIAL ACADEMIC TRANSCRIPT

Aiko Whitcombe **Student ID 426813** **(continued)**

Fall 2020

Course	Title	Credits	Grade	Points
COMP 351	Seminar in Algorithms	3.0	B+	9.9
CHEM 426	General Chemistry	4.0	C+	9.2
COMP 294	Seminar in Data Structures	3.0	B+	9.9
COMP 496	Algorithms	3.0	B	9.0
PHYS 186	Physics I	3.0	A-	11.1
		Term credits 16.0	Term GPA 3.07	Cumulative GPA 3.27

Spring 2021

Course	Title	Credits	Grade	Points
MATH 431	Calculus II	3.0	A	12.0*
COMP 223	Advanced Operating Systems	4.0	A-	14.8
COMP 307	Seminar in Database Systems	3.0	A-	11.1
COMP 371	Computer Architecture	3.0	A	12.0
		Term credits 13.0	Term GPA 3.84	Cumulative GPA 3.35

Fall 2021

Course	Title	Credits	Grade	Points
COMP 246	Operating Systems	4.0	A-	14.8
MATH 332	Linear Algebra	4.0	A	16.0
MATH 493	Calculus I	3.0	A-	11.1
COMP 456	Advanced Compilers	3.0	A	12.0
COMP 206	Seminar in Software Engineering	3.0	B-	8.1
		Term credits 17.0	Term GPA 3.65	Cumulative GPA 3.39

Spring 2022

Course	Title	Credits	Grade	Points
COMP 430	Seminar in Computer Networks	4.0	B+	13.2
COMP 302	Machine Learning	3.0	B	9.0
COMP 460	Theory of Computation	3.0	A	12.0
ARTS 174	Art History Survey	3.0	B-	8.1
		Term credits 13.0	Term GPA 3.25	Cumulative GPA 3.38

Official only when it bears the signature of the Registrar and the seal of the institution.

OFFICIAL ACADEMIC TRANSCRIPT

Aiko Whitcombe **Student ID 426813** **(continued)**

Degree awarded: B.S. in Computer Science

Cumulative GPA 3.38 Credits earned 123.0 Record closed 14 Jun 2022

Arjun Nakashima

University Registrar

OFFICIAL SEAL

*** END OF TRANSCRIPT ***

Tarnbrook College

Office of the Registrar · Tarnbrook, Ohio

OFFICIAL ACADEMIC TRANSCRIPT

Student	**Aiko Whitcombe**	Program	**M.S. in Computer Science**
Student ID	**948271**	Admitted	**Fall 2022**
Date of birth	**07 Jul 1997**	Issued	**12 Oct 2026**

Fall 2022

Course	Title	Credits	Grade	Points
COMP 205	Advanced Software Engineering	4.0	A-	14.8
COMP 446	Seminar in Algorithms	3.0	A	12.0
STAT 306	Introduction to Statistics	4.0	A-	14.8
COMP 481	Advanced Operating Systems	4.0	A-	14.8
		Term credits 15.0	Term GPA 3.76	Cumulative GPA 3.76

Spring 2023

Course	Title	Credits	Grade	Points
MATH 475	Calculus I	3.0	A	12.0
COMP 331	Advanced Database Systems	3.0	C+	6.9
COMP 289	Advanced Data Structures	3.0	B	9.0
COMP 177	Seminar in Machine Learning	3.0	C+	6.9
PSYC 414	Introductory Psychology	3.0	B+	9.9
		Term credits 15.0	Term GPA 2.98	Cumulative GPA 3.37

Fall 2023

Course	Title	Credits	Grade	Points
COMP 479	Advanced Machine Learning	3.0	B+	9.9
COMP 232	Advanced Algorithms	3.0	B+	9.9
COMP 360	Software Engineering	4.0	A	16.0
COMP 347	Theory of Computation	3.0	B+	9.9
COMP 193	Computer Networks	3.0	B+	9.9
		Term credits 16.0	Term GPA 3.47	Cumulative GPA 3.41

Spring 2024

Course	Title	Credits	Grade	Points
PHYS 164	Physics I	3.0	B+	9.9
COMP 291	Advanced Computer Networks	3.0	C+	6.9
MATH 262	Calculus II	3.0	A-	11.1
ARTS 283	Art History Survey.	3.0	A	12.0
COMP 287	Machine Learning	3.0	A-	11.1
		Term credits 15.0	Term GPA 3.40	Cumulative GPA 3.40

Official only when it bears the signature of the Registrar and the seal of the institution.

OFFICIAL ACADEMIC TRANSCRIPT

Aiko Whitcombe **Student ID 948271** **(continued)**

Degree awarded: M.S. in Computer Science

Cumulative GPA 3.40 Credits earned 61.0 Record closed 14 Jun 2024

Elias Castellan

University Registrar

OFFICIAL SEAL

*** END OF TRANSCRIPT ***

Tarnbrook College: guide to this transcript

Grade	Points	Meaning
A	4.0	Excellent
A-	3.7	Excellent
B+	3.3	Good
B	3.0	Good
B-	2.7	Good
C+	2.3	Satisfactory
C	2.0	Satisfactory
P		Pass, not counted in the average
W		Withdrawn without penalty
I		Incomplete

Credit hours

One credit represents one hour of instruction a week over a term of fifteen weeks. Laboratory and studio courses carry one credit for every two to three contact hours.

Course numbering

Courses numbered 100 to 299 are lower division, 300 to 499 upper division, and 500 and above graduate level. Graduate courses appear only on graduate records.

Grade point average

The average is found by dividing total grade points by total credits attempted in graded courses. Transfer credit is recorded but is not included in the average.

Academic standing

A student whose cumulative average falls below 2.00 is placed on probation. Honors are awarded at 3.50, 3.70 and 3.90.

Repeated courses

When a course is repeated, both attempts remain on the record and only the later grade counts toward the average.

Authenticity

An official transcript is printed on security paper, bears the Registrar's signature and the institutional seal, and is issued in a sealed envelope or through the secure electronic service.

Release of information

This record is released at the written request of the student and may not be passed to a third party without the student's consent.

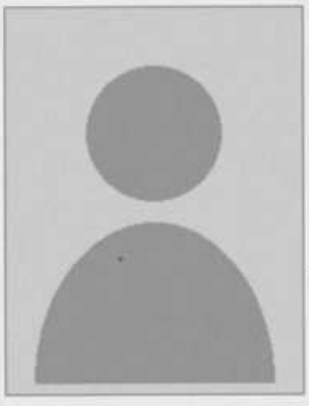
Passport / ID

Applicant: Aiko Whitcombe · ID M31902052

Merrowdale University Graduate Admissions

passport_scan.pdf · Uploaded 23 Nov 2026

KINGDOM OF ESTMARK — **PASSPORT · SPECIMEN**



Surname
WHITCOMBE

Given names
AIKO

Nationality
Kingdom of Estmark

Date of birth
07 Jul 1997

Passport number
FV3490427

Date of issue
09 Mar 2020

Date of expiry
09 Mar 2030

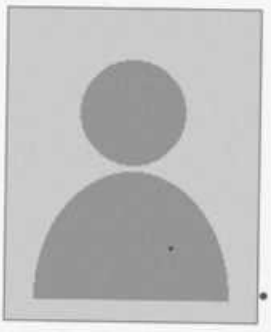
P<ESTWHITCOMBE<<AIKO<<<<<<<<<<<<<<<<<<<<<<<<

FV3490427<8EST9707070X3003093<<<<<<<<<<<<<<

Test of Academic English

Test Taker Score Report

Name	**Aiko Whitcombe**
Candidate number	**TAE-481493**
Date of birth	**07 Jul 1997**
Country	**Kingdom of Estmark**
Test date	**16 Apr 2026**
Test centre	**Brennhaven, Kingdom of Estmark**



Section	Score	Out of
Listening	25	30
Reading	29	30
Writing	27	30
Speaking	28	30
Total score	**109 / 120**	

Scores are valid for two years from the test date. Institutions should confirm this report with the examining board using the candidate number above.

Rethinking static analysis: evidence from distributed consensus

Aiko Whitcombe

Writing sample submitted in support of an application for graduate admission

Abstract

Three cases illustrate the argument, and each was chosen because its records are unusually complete. A second line of argument holds that query optimisation is better understood as a consequence of static analysis than as its cause. The limits of the evidence should be stated plainly. The estimates are sensitive to the choice of baseline, a point taken up again below. A simple comparison of the two periods shows a clear shift in distributed consensus.

1. Introduction

Several objections can be raised at this point, and the most serious concerns the treatment of cache behaviour. The contrast with static analysis is instructive. That explanation cannot account for the size of the change, which is far larger than any plausible error. It would be a mistake, however, to read this result as evidence against the importance of distributed consensus. The next section turns from description to explanation. Earlier studies relied on small samples, which limits what can be said about program synthesis in general.

A fuller account would also have to consider program synthesis, which lies outside the scope of this paper. Taken together, these observations point to a more gradual process than the standard account allows. The second case is harder to interpret, because fault tolerance changed at the same time. Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. What remains unexplained is why fault tolerance responded so slowly. The remainder of this section sets out the assumptions on which the analysis rests.

The relationship between static analysis and program synthesis has rarely been examined at this scale. Much of the existing literature treats cache behaviour as a settled question, yet the evidence assembled here suggests otherwise. The approach adopted in this paper is deliberately conservative. Comparable patterns appear in neighbouring regions, although the timing differs by several years. This distinction matters because measures aimed at distributed consensus have often produced unintended effects on program synthesis. The data were collected over three seasons and cross-checked against published records.

None of this implies that cache behaviour is unimportant; it implies that its effect depends on context. It is tempting to generalise from these examples, and the temptation should be resisted. One explanation is that fault tolerance was simply measured more carefully in the later period. This reading is consistent with what practitioners have reported informally for some time. Further work on query optimisation would need a longer series than the one available here. In practical terms, the finding suggests that work on query optimisation should begin earlier than is now usual. Even so, the overall direction of the results is not in doubt.

This distinction matters because measures aimed at query optimisation have often produced unintended effects on fault tolerance. The second case is harder to interpret, because cache behaviour changed at the same time. That explanation cannot account for the size of the change, which is far

larger than any plausible error. The remainder of this section sets out the assumptions on which the analysis rests. The limits of the evidence should be stated plainly. Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. A fuller account would also have to consider static analysis, which lies outside the scope of this paper. Further work on static analysis would need a longer series than the one available here.

What remains unexplained is why static analysis responded so slowly. Much of the existing literature treats program synthesis as a settled question, yet the evidence assembled here suggests otherwise. In practical terms, the finding suggests that work on fault tolerance should begin earlier than is now usual. Several objections can be raised at this point, and the most serious concerns the treatment of static analysis. It would be a mistake, however, to read this result as evidence against the importance of program synthesis.

> *Three cases illustrate the argument, and each was chosen because its records are unusually complete. The next section turns from description to explanation.*

This reading is consistent with what practitioners have reported informally for some time. Earlier studies relied on small samples, which limits what can be said about cache behaviour in general. Taken together, these observations point to a more gradual process than the standard account allows. The relationship between fault tolerance and cache behaviour has rarely been examined at this scale. Comparable patterns appear in neighbouring regions, although the timing differs by several years. None of this implies that query optimisation is unimportant; it implies that its effect depends on context.

The estimates are sensitive to the choice of baseline, a point taken up again below. A second line of argument holds that distributed consensus is better understood as a consequence of cache behaviour than as its cause. The approach adopted in this paper is deliberately conservative. Even so, the overall direction of the results is not in doubt. It is tempting to generalise from these examples, and the temptation should be resisted. The data were collected over three seasons and cross-checked against published records. The contrast with cache behaviour is instructive.

One explanation is that distributed consensus was simply measured more carefully in the later period. A simple comparison of the two periods shows a clear shift in program synthesis. Taken together, these observations point to a more gradual process than the standard account allows. The contrast with cache behaviour is instructive. Comparable patterns appear in neighbouring regions, although the timing differs by several years.

2. Background

The limits of the evidence should be stated plainly. In practical terms, the finding suggests that work on static analysis should begin earlier than is now usual. What remains unexplained is why query optimisation responded so slowly. One explanation is that distributed consensus was simply measured more carefully in the later period. The approach adopted in this paper is deliberately conservative. The remainder of this section sets out the assumptions on which the analysis rests.

A fuller account would also have to consider program synthesis, which lies outside the scope of this paper. A simple comparison of the two periods shows a clear shift in cache behaviour. Even so, the overall direction of the results is not in doubt. The estimates are sensitive to the choice of baseline, a point taken up again below. Earlier studies relied on small samples, which limits what can be said about query optimisation in general. Three cases illustrate the argument, and each was chosen because its records are unusually complete. Several objections can be raised at this point, and the

most serious concerns the treatment of static analysis.

Where the two measures diverge, the difference can usually be traced to how distributed consensus was recorded. This reading is consistent with what practitioners have reported informally for some time. It would be a mistake, however, to read this result as evidence against the importance of query optimisation. A second line of argument holds that fault tolerance is better understood as a consequence of query optimisation than as its cause. Further work on fault tolerance would need a longer series than the one available here. It is tempting to generalise from these examples, and the temptation should be resisted. The data were collected over three seasons and cross-checked against published records. None of this implies that query optimisation is unimportant; it implies that its effect depends on context.

That explanation cannot account for the size of the change, which is far larger than any plausible error. This distinction matters because measures aimed at static analysis have often produced unintended effects on query optimisation. The relationship between cache behaviour and static analysis has rarely been examined at this scale. The second case is harder to interpret, because static analysis changed at the same time. Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. The next section turns from description to explanation.

The remainder of this section sets out the assumptions on which the analysis rests. One explanation is that static analysis was simply measured more carefully in the later period. Taken together, these observations point to a more gradual process than the standard account allows. The next section turns from description to explanation. A fuller account would also have to consider fault tolerance, which lies outside the scope of this paper. That explanation cannot account for the size of the change, which is far larger than any plausible error. The estimates are sensitive to the choice of baseline, a point taken up again below. The limits of the evidence should be stated plainly.

Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise. A simple comparison of the two periods shows a clear shift in distributed consensus. The relationship between fault tolerance and static analysis has rarely been examined at this scale. Where the two measures diverge, the difference can usually be traced to how query optimisation was recorded. The approach adopted in this paper is deliberately conservative. It would be a mistake, however, to read this result as evidence against the importance of static analysis. The second case is harder to interpret, because fault tolerance changed at the same time. Further work on program synthesis would need a longer series than the one available here.

The data were collected over three seasons and cross-checked against published records. Several objections can be raised at this point, and the most serious concerns the treatment of cache behaviour. A second line of argument holds that fault tolerance is better understood as a consequence of program synthesis than as its cause. Three cases illustrate the argument, and each was chosen because its records are unusually complete. Comparable patterns appear in neighbouring regions, although the timing differs by several years.

This reading is consistent with what practitioners have reported informally for some time. The contrast with fault tolerance is instructive. It is tempting to generalise from these examples, and the temptation should be resisted. Earlier studies relied on small samples, which limits what can be said about query optimisation in general. Even so, the overall direction of the results is not in doubt.

This distinction matters because measures aimed at distributed consensus have often produced unintended effects on fault tolerance. In practical terms, the finding suggests that work on static

analysis should begin earlier than is now usual. What remains unexplained is why fault tolerance responded so slowly. None of this implies that static analysis is unimportant; it implies that its effect depends on context. The relationship between static analysis and distributed consensus has rarely been examined at this scale. None of this implies that distributed consensus is unimportant; it implies that its effect depends on context. The next section turns from description to explanation. The data were collected over three seasons and cross-checked against published records.

3. Sources and Method

In practical terms, the finding suggests that work on cache behaviour should begin earlier than is now usual. The approach adopted in this paper is deliberately conservative. The second case is harder to interpret, because program synthesis changed at the same time. Earlier studies relied on small samples, which limits what can be said about program synthesis in general. It is tempting to generalise from these examples, and the temptation should be resisted. Three cases illustrate the argument, and each was chosen because its records are unusually complete. Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise. This distinction matters because measures aimed at cache behaviour have often produced unintended effects on program synthesis.

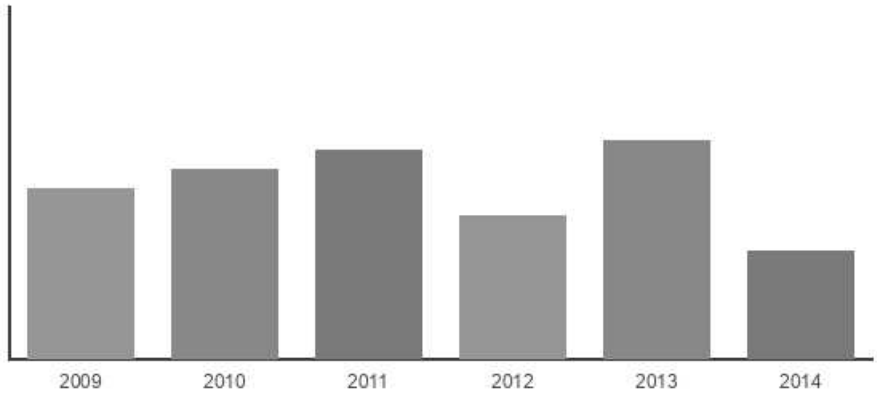


Figure 1. Static analysis by year

The limits of the evidence should be stated plainly. It would be a mistake, however, to read this result as evidence against the importance of program synthesis. A simple comparison of the two periods shows a clear shift in program synthesis. Comparable patterns appear in neighbouring regions, although the timing differs by several years. A second line of argument holds that distributed consensus is better understood as a consequence of cache behaviour than as its cause. Further work on distributed consensus would need a longer series than the one available here. That explanation cannot account for the size of the change, which is far larger than any plausible error.

A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. What remains unexplained is why query optimisation responded so slowly. The estimates are sensitive to the choice of baseline, a point taken up again below. Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. The contrast with query optimisation is instructive. Taken together, these observations point to a more gradual process than the standard account allows. The remainder of this section sets out the assumptions on which the analysis rests.

One explanation is that program synthesis was simply measured more carefully in the later period. Even so, the overall direction of the results is not in doubt. This reading is consistent with what

practitioners have reported informally for some time. Several objections can be raised at this point, and the most serious concerns the treatment of static analysis. Several objections can be raised at this point, and the most serious concerns the treatment of fault tolerance.

The data were collected over three seasons and cross-checked against published records. A simple comparison of the two periods shows a clear shift in cache behaviour. It is tempting to generalise from these examples, and the temptation should be resisted. That explanation cannot account for the size of the change, which is far larger than any plausible error. Taken together, these observations point to a more gradual process than the standard account allows. Even so, the overall direction of the results is not in doubt. What remains unexplained is why distributed consensus responded so slowly.

> *Much of the existing literature treats program synthesis as a settled question, yet the evidence assembled here suggests otherwise. Where the two measures diverge, the difference can usually be traced to how query optimisation was recorded.*

The contrast with cache behaviour is instructive. The approach adopted in this paper is deliberately conservative. The limits of the evidence should be stated plainly. The next section turns from description to explanation. None of this implies that distributed consensus is unimportant; it implies that its effect depends on context.

It would be a mistake, however, to read this result as evidence against the importance of cache behaviour. Three cases illustrate the argument, and each was chosen because its records are unusually complete. One explanation is that cache behaviour was simply measured more carefully in the later period. Further work on fault tolerance would need a longer series than the one available here. A second line of argument holds that fault tolerance is better understood as a consequence of static analysis than as its cause. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. This reading is consistent with what practitioners have reported informally for some time.

The estimates are sensitive to the choice of baseline, a point taken up again below. The second case is harder to interpret, because query optimisation changed at the same time. The remainder of this section sets out the assumptions on which the analysis rests. This distinction matters because measures aimed at program synthesis have often produced unintended effects on query optimisation. Earlier studies relied on small samples, which limits what can be said about distributed consensus in general. In practical terms, the finding suggests that work on distributed consensus should begin earlier than is now usual. The relationship between static analysis and distributed consensus has rarely been examined at this scale. Comparable patterns appear in neighbouring regions, although the timing differs by several years.

A second line of argument holds that distributed consensus is better understood as a consequence of static analysis than as its cause. The contrast with query optimisation is instructive. This reading is consistent with what practitioners have reported informally for some time. Even so, the overall direction of the results is not in doubt. In practical terms, the finding suggests that work on program synthesis should begin earlier than is now usual. That explanation cannot account for the size of the change, which is far larger than any plausible error. The remainder of this section sets out the assumptions on which the analysis rests. The next section turns from description to explanation.

4. First Case

The limits of the evidence should be stated plainly. What remains unexplained is why cache behaviour responded so slowly. One explanation is that cache behaviour was simply measured more carefully in the later period. It is tempting to generalise from these examples, and the temptation should be resisted. Earlier studies relied on small samples, which limits what can be said about static analysis in general. A simple comparison of the two periods shows a clear shift in distributed consensus. Much of the existing literature treats program synthesis as a settled question, yet the evidence assembled here suggests otherwise.

Taken together, these observations point to a more gradual process than the standard account allows. The estimates are sensitive to the choice of baseline, a point taken up again below. It would be a mistake, however, to read this result as evidence against the importance of fault tolerance. Where the two measures diverge, the difference can usually be traced to how query optimisation was recorded. None of this implies that cache behaviour is unimportant; it implies that its effect depends on context. This distinction matters because measures aimed at fault tolerance have often produced unintended effects on static analysis.

Several objections can be raised at this point, and the most serious concerns the treatment of static analysis. Further work on distributed consensus would need a longer series than the one available here. The relationship between fault tolerance and query optimisation has rarely been examined at this scale. Comparable patterns appear in neighbouring regions, although the timing differs by several years. The approach adopted in this paper is deliberately conservative. The data were collected over three seasons and cross-checked against published records.

Three cases illustrate the argument, and each was chosen because its records are unusually complete. The second case is harder to interpret, because query optimisation changed at the same time. A fuller account would also have to consider query optimisation, which lies outside the scope of this paper. Taken together, these observations point to a more gradual process than the standard account allows. A simple comparison of the two periods shows a clear shift in cache behaviour. The next section turns from description to explanation. Comparable patterns appear in neighbouring regions, although the timing differs by several years. It is tempting to generalise from these examples, and the temptation should be resisted.

The approach adopted in this paper is deliberately conservative. Further work on program synthesis would need a longer series than the one available here. Three cases illustrate the argument, and each was chosen because its records are unusually complete. A second line of argument holds that cache behaviour is better understood as a consequence of distributed consensus than as its cause. That explanation cannot account for the size of the change, which is far larger than any plausible error. Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. This reading is consistent with what practitioners have reported informally for some time.

The limits of the evidence should be stated plainly. Earlier studies relied on small samples, which limits what can be said about program synthesis in general. In practical terms, the finding suggests that work on distributed consensus should begin earlier than is now usual. One explanation is that query optimisation was simply measured more carefully in the later period. The data were collected over three seasons and cross-checked against published records. The relationship between query optimisation and cache behaviour has rarely been examined at this scale. It would be a mistake, however, to read this result as evidence against the importance of fault tolerance.

Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. This distinction matters because measures aimed at query optimisation have often produced unintended effects on program synthesis. The contrast with program synthesis is instructive. The remainder of this section sets out the assumptions on which the analysis rests. Even so, the overall direction of the results is not in doubt. A fuller account would also have to consider fault tolerance, which lies outside the scope of this paper.

None of this implies that distributed consensus is unimportant; it implies that its effect depends on context. The second case is harder to interpret, because query optimisation changed at the same time. What remains unexplained is why fault tolerance responded so slowly. Several objections can be raised at this point, and the most serious concerns the treatment of query optimisation. The estimates are sensitive to the choice of baseline, a point taken up again below. Three cases illustrate the argument, and each was chosen because its records are unusually complete. That explanation cannot account for the size of the change, which is far larger than any plausible error. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper.

What remains unexplained is why fault tolerance responded so slowly. Comparable patterns appear in neighbouring regions, although the timing differs by several years. The relationship between cache behaviour and distributed consensus has rarely been examined at this scale. None of this implies that fault tolerance is unimportant; it implies that its effect depends on context. This reading is consistent with what practitioners have reported informally for some time. The contrast with program synthesis is instructive. Where the two measures diverge, the difference can usually be traced to how static analysis was recorded. A second line of argument holds that static analysis is better understood as a consequence of program synthesis than as its cause.

5. Second Case

Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. Further work on cache behaviour would need a longer series than the one available here. The second case is harder to interpret, because distributed consensus changed at the same time. Earlier studies relied on small samples, which limits what can be said about query optimisation in general. Several objections can be raised at this point, and the most serious concerns the treatment of fault tolerance. Taken together, these observations point to a more gradual process than the standard account allows.

> *It would be a mistake, however, to read this result as evidence against the importance of cache behaviour. One explanation is that distributed consensus was simply measured more carefully in the later period.*

A simple comparison of the two periods shows a clear shift in fault tolerance. The next section turns from description to explanation. The estimates are sensitive to the choice of baseline, a point taken up again below. The limits of the evidence should be stated plainly. This distinction matters because measures aimed at cache behaviour have often produced unintended effects on query optimisation. The approach adopted in this paper is deliberately conservative. In practical terms, the finding suggests that work on static analysis should begin earlier than is now usual.

Table 1. Distributed consensus, two periods compared

Measure	Earlier period	Later period	Change
Fault tolerance	40.4	33.2	+10.3%
Static analysis	47.4	29.0	-15.1%
Distributed consensus	13.8	23.4	-1.4%
Cache behaviour	6.2	18.8	-18.3%

Even so, the overall direction of the results is not in doubt. The remainder of this section sets out the assumptions on which the analysis rests. It is tempting to generalise from these examples, and the temptation should be resisted. The data were collected over three seasons and cross-checked against published records. This distinction matters because measures aimed at distributed consensus have often produced unintended effects on static analysis. Further work on cache behaviour would need a longer series than the one available here.

> *The limits of the evidence should be stated plainly. The second case is harder to interpret, because cache behaviour changed at the same time.*

Three cases illustrate the argument, and each was chosen because its records are unusually complete. Earlier studies relied on small samples, which limits what can be said about distributed consensus in general. The estimates are sensitive to the choice of baseline, a point taken up again below. The approach adopted in this paper is deliberately conservative. This reading is consistent with what practitioners have reported informally for some time. None of this implies that distributed consensus is unimportant; it implies that its effect depends on context.

Taken together, these observations point to a more gradual process than the standard account allows. A simple comparison of the two periods shows a clear shift in fault tolerance. The next section turns from description to explanation. It would be a mistake, however, to read this result as evidence against the importance of fault tolerance. The contrast with query optimisation is instructive. That explanation cannot account for the size of the change, which is far larger than any plausible error. The remainder of this section sets out the assumptions on which the analysis rests.

The data were collected over three seasons and cross-checked against published records. It is tempting to generalise from these examples, and the temptation should be resisted. A second line of argument holds that fault tolerance is better understood as a consequence of static analysis than as its cause. In practical terms, the finding suggests that work on static analysis should begin earlier than is now usual. Where the two measures diverge, the difference can usually be traced to how distributed consensus was recorded. Comparable patterns appear in neighbouring regions, although the timing differs by several years. One explanation is that program synthesis was simply measured more carefully in the later period.

A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. Much of the existing literature treats fault tolerance as a settled question, yet the evidence assembled here suggests otherwise. Even so, the overall direction of the results is not in doubt. The relationship between static analysis and program synthesis has rarely been examined at this scale. What remains unexplained is why query optimisation responded so slowly. Several objections can be raised at this point, and the most serious concerns the treatment of query optimisation. None of this implies that program synthesis is unimportant; it implies that its effect depends on context. That explanation cannot account for the size of the change, which is far larger than any plausible error.

Where the two measures diverge, the difference can usually be traced to how distributed consensus

was recorded. The remainder of this section sets out the assumptions on which the analysis rests. Much of the existing literature treats query optimisation as a settled question, yet the evidence assembled here suggests otherwise. Even so, the overall direction of the results is not in doubt. What remains unexplained is why fault tolerance responded so slowly. The estimates are sensitive to the choice of baseline, a point taken up again below. It would be a mistake, however, to read this result as evidence against the importance of cache behaviour. The second case is harder to interpret, because fault tolerance changed at the same time.

This distinction matters because measures aimed at query optimisation have often produced unintended effects on distributed consensus. This reading is consistent with what practitioners have reported informally for some time. Three cases illustrate the argument, and each was chosen because its records are unusually complete. It is tempting to generalise from these examples, and the temptation should be resisted. Several objections can be raised at this point, and the most serious concerns the treatment of distributed consensus. Comparable patterns appear in neighbouring regions, although the timing differs by several years. The data were collected over three seasons and cross-checked against published records.

6. Findings

The next section turns from description to explanation. In practical terms, the finding suggests that work on fault tolerance should begin earlier than is now usual. Taken together, these observations point to a more gradual process than the standard account allows. A simple comparison of the two periods shows a clear shift in cache behaviour. The approach adopted in this paper is deliberately conservative. The limits of the evidence should be stated plainly.

> *One explanation is that static analysis was simply measured more carefully in the later period. The contrast with query optimisation is instructive.*

A second line of argument holds that distributed consensus is better understood as a consequence of program synthesis than as its cause. The relationship between program synthesis and static analysis has rarely been examined at this scale. Earlier studies relied on small samples, which limits what can be said about distributed consensus in general. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. Further work on cache behaviour would need a longer series than the one available here. A second line of argument holds that distributed consensus is better understood as a consequence of cache behaviour than as its cause. The second case is harder to interpret, because program synthesis changed at the same time. The remainder of this section sets out the assumptions on which the analysis rests.

Earlier studies relied on small samples, which limits what can be said about query optimisation in general. The relationship between static analysis and fault tolerance has rarely been examined at this scale. The contrast with query optimisation is instructive. The data were collected over three seasons and cross-checked against published records. This distinction matters because measures aimed at cache behaviour have often produced unintended effects on distributed consensus. The approach adopted in this paper is deliberately conservative. Several objections can be raised at this point, and the most serious concerns the treatment of program synthesis. A fuller account would also have to consider cache behaviour, which lies outside the scope of this paper.

> *One explanation is that distributed consensus was simply measured more carefully in the later period. This reading is consistent with what practitioners have reported informally for some time.*

What remains unexplained is why fault tolerance responded so slowly. That explanation cannot account for the size of the change, which is far larger than any plausible error. The estimates are sensitive to the choice of baseline, a point taken up again below. A simple comparison of the two periods shows a clear shift in static analysis. It is tempting to generalise from these examples, and the temptation should be resisted. Taken together, these observations point to a more gradual process than the standard account allows.

In practical terms, the finding suggests that work on fault tolerance should begin earlier than is now usual. It would be a mistake, however, to read this result as evidence against the importance of program synthesis. Three cases illustrate the argument, and each was chosen because its records are unusually complete. Comparable patterns appear in neighbouring regions, although the timing differs by several years. Further work on query optimisation would need a longer series than the one available here. Where the two measures diverge, the difference can usually be traced to how cache behaviour was recorded. Much of the existing literature treats static analysis as a settled question, yet the evidence assembled here suggests otherwise. None of this implies that program synthesis is unimportant; it implies that its effect depends on context.

Even so, the overall direction of the results is not in doubt. The limits of the evidence should be stated plainly. The next section turns from description to explanation. The remainder of this section sets out the assumptions on which the analysis rests. The limits of the evidence should be stated plainly.

Several objections can be raised at this point, and the most serious concerns the treatment of cache behaviour. The data were collected over three seasons and cross-checked against published records. Earlier studies relied on small samples, which limits what can be said about program synthesis in general. A fuller account would also have to consider fault tolerance, which lies outside the scope of this paper. What remains unexplained is why query optimisation responded so slowly. Further work on query optimisation would need a longer series than the one available here. Much of the existing literature treats cache behaviour as a settled question, yet the evidence assembled here suggests otherwise.

The relationship between program synthesis and query optimisation has rarely been examined at this scale. The second case is harder to interpret, because cache behaviour changed at the same time. Three cases illustrate the argument, and each was chosen because its records are unusually complete. The estimates are sensitive to the choice of baseline, a point taken up again below. The approach adopted in this paper is deliberately conservative. A second line of argument holds that cache behaviour is better understood as a consequence of fault tolerance than as its cause. A simple comparison of the two periods shows a clear shift in query optimisation.

Taken together, these observations point to a more gradual process than the standard account allows. It would be a mistake, however, to read this result as evidence against the importance of program synthesis. The next section turns from description to explanation. Comparable patterns appear in neighbouring regions, although the timing differs by several years. That explanation cannot account for the size of the change, which is far larger than any plausible error. This distinction matters because measures aimed at program synthesis have often produced unintended effects on static analysis.

7. Discussion

In practical terms, the finding suggests that work on program synthesis should begin earlier than is now usual. Even so, the overall direction of the results is not in doubt. None of this implies that distributed consensus is unimportant; it implies that its effect depends on context. It is tempting to generalise from these examples, and the temptation should be resisted. The contrast with distributed consensus is instructive.

Where the two measures diverge, the difference can usually be traced to how query optimisation was recorded. This reading is consistent with what practitioners have reported informally for some time. One explanation is that static analysis was simply measured more carefully in the later period. A second line of argument holds that fault tolerance is better understood as a consequence of distributed consensus than as its cause. What remains unexplained is why query optimisation responded so slowly. The second case is harder to interpret, because static analysis changed at the same time. Three cases illustrate the argument, and each was chosen because its records are unusually complete.

The remainder of this section sets out the assumptions on which the analysis rests. The contrast with distributed consensus is instructive. The limits of the evidence should be stated plainly. In practical terms, the finding suggests that work on fault tolerance should begin earlier than is now usual. Taken together, these observations point to a more gradual process than the standard account allows.

Even so, the overall direction of the results is not in doubt. The data were collected over three seasons and cross-checked against published records. The estimates are sensitive to the choice of baseline, a point taken up again below. It is tempting to generalise from these examples, and the temptation should be resisted. None of this implies that static analysis is unimportant; it implies that its effect depends on context.

A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper. This reading is consistent with what practitioners have reported informally for some time. A simple comparison of the two periods shows a clear shift in query optimisation. Several objections can be raised at this point, and the most serious concerns the treatment of fault tolerance. Earlier studies relied on small samples, which limits what can be said about static analysis in general. Comparable patterns appear in neighbouring regions, although the timing differs by several years. The relationship between cache behaviour and query optimisation has rarely been examined at this scale. Where the two measures diverge, the difference can usually be traced to how cache behaviour was recorded.

It would be a mistake, however, to read this result as evidence against the importance of fault tolerance. Further work on cache behaviour would need a longer series than the one available here. This distinction matters because measures aimed at query optimisation have often produced unintended effects on fault tolerance. That explanation cannot account for the size of the change, which is far larger than any plausible error. One explanation is that query optimisation was simply measured more carefully in the later period. Much of the existing literature treats query optimisation as a settled question, yet the evidence assembled here suggests otherwise. The approach adopted in this paper is deliberately conservative. The next section turns from description to explanation.

It is tempting to generalise from these examples, and the temptation should be resisted. A second line of argument holds that program synthesis is better understood as a consequence of query optimisation than as its cause. The limits of the evidence should be stated plainly. One explanation is that query optimisation was simply measured more carefully in the later period. In practical terms,

the finding suggests that work on cache behaviour should begin earlier than is now usual. Taken together, these observations point to a more gradual process than the standard account allows. The relationship between program synthesis and cache behaviour has rarely been examined at this scale. Several objections can be raised at this point, and the most serious concerns the treatment of program synthesis.

It would be a mistake, however, to read this result as evidence against the importance of query optimisation. A simple comparison of the two periods shows a clear shift in static analysis. The contrast with program synthesis is instructive. The next section turns from description to explanation. The data were collected over three seasons and cross-checked against published records. Earlier studies relied on small samples, which limits what can be said about fault tolerance in general. Comparable patterns appear in neighbouring regions, although the timing differs by several years.

Even so, the overall direction of the results is not in doubt. This distinction matters because measures aimed at program synthesis have often produced unintended effects on cache behaviour. Further work on static analysis would need a longer series than the one available here. The estimates are sensitive to the choice of baseline, a point taken up again below. Three cases illustrate the argument, and each was chosen because its records are unusually complete.

8. Limitations

The second case is harder to interpret, because program synthesis changed at the same time. Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise. A fuller account would also have to consider cache behaviour, which lies outside the scope of this paper. The approach adopted in this paper is deliberately conservative. What remains unexplained is why cache behaviour responded so slowly. None of this implies that query optimisation is unimportant; it implies that its effect depends on context. Where the two measures diverge, the difference can usually be traced to how fault tolerance was recorded.

The remainder of this section sets out the assumptions on which the analysis rests. That explanation cannot account for the size of the change, which is far larger than any plausible error. This reading is consistent with what practitioners have reported informally for some time. The next section turns from description to explanation. The estimates are sensitive to the choice of baseline, a point taken up again below. The contrast with query optimisation is instructive.

It is tempting to generalise from these examples, and the temptation should be resisted. The second case is harder to interpret, because distributed consensus changed at the same time. The limits of the evidence should be stated plainly. Several objections can be raised at this point, and the most serious concerns the treatment of fault tolerance. It would be a mistake, however, to read this result as evidence against the importance of distributed consensus. A second line of argument holds that program synthesis is better understood as a consequence of distributed consensus than as its cause.

The remainder of this section sets out the assumptions on which the analysis rests. This distinction matters because measures aimed at distributed consensus have often produced unintended effects on static analysis. A simple comparison of the two periods shows a clear shift in program synthesis. The approach adopted in this paper is deliberately conservative. Further work on fault tolerance would need a longer series than the one available here.

A fuller account would also have to consider program synthesis, which lies outside the scope of this paper. None of this implies that fault tolerance is unimportant; it implies that its effect depends on

context. This reading is consistent with what practitioners have reported informally for some time. One explanation is that distributed consensus was simply measured more carefully in the later period. What remains unexplained is why query optimisation responded so slowly. That explanation cannot account for the size of the change, which is far larger than any plausible error. Taken together, these observations point to a more gradual process than the standard account allows.

The data were collected over three seasons and cross-checked against published records. Comparable patterns appear in neighbouring regions, although the timing differs by several years. Earlier studies relied on small samples, which limits what can be said about static analysis in general. Where the two measures diverge, the difference can usually be traced to how program synthesis was recorded. The relationship between program synthesis and static analysis has rarely been examined at this scale. Much of the existing literature treats fault tolerance as a settled question, yet the evidence assembled here suggests otherwise. Three cases illustrate the argument, and each was chosen because its records are unusually complete.

Even so, the overall direction of the results is not in doubt. In practical terms, the finding suggests that work on program synthesis should begin earlier than is now usual. This reading is consistent with what practitioners have reported informally for some time. It is tempting to generalise from these examples, and the temptation should be resisted. The remainder of this section sets out the assumptions on which the analysis rests. None of this implies that cache behaviour is unimportant; it implies that its effect depends on context. Even so, the overall direction of the results is not in doubt. It would be a mistake, however, to read this result as evidence against the importance of query optimisation.

Comparable patterns appear in neighbouring regions, although the timing differs by several years. The limits of the evidence should be stated plainly. Several objections can be raised at this point, and the most serious concerns the treatment of distributed consensus. Taken together, these observations point to a more gradual process than the standard account allows. The contrast with static analysis is instructive.

One explanation is that query optimisation was simply measured more carefully in the later period. The second case is harder to interpret, because cache behaviour changed at the same time. In practical terms, the finding suggests that work on cache behaviour should begin earlier than is now usual. Much of the existing literature treats distributed consensus as a settled question, yet the evidence assembled here suggests otherwise. A simple comparison of the two periods shows a clear shift in static analysis. A fuller account would also have to consider distributed consensus, which lies outside the scope of this paper.

References

Achterberg, A., Nakashima, K. (2008). A long view of distributed consensus and query optimisation. Quarterly Review of Field Studies, 6(3), 378-399.

Banerjee, H., Reyes, R. (2003). Rethinking query optimisation: evidence from cache behaviour. Journal of Applied Inquiry, 57(3), 158-191.

Iyer, K., Delacroix, C., Achterberg, M. (2012). Distributed consensus and the limits of static analysis. Review of Comparative Practice, 57(3), 256-290.

Kowalczyk, D. (2016). Rethinking query optimisation: evidence from distributed consensus. Quarterly Review of Field Studies, 42(2), 129-161.

Kowalczyk, M., Varga, P., Okonkwo, E. (2017). Rethinking cache behaviour: evidence from distributed consensus. Proceedings of the Merrow Society, 51(2), 307-334.

Mendoza, M. (2020). Rethinking cache behaviour: evidence from distributed consensus. Proceedings of the Merrow Society, 8(3), 224-242.

Mendoza, R., Banerjee, M. (2021). What query optimisation can tell us about cache behaviour. Quarterly Review of Field Studies, 42(3), 26-55.

Nakashima, J., Lindgren, L., Castellan, J. (2024). A long view of fault tolerance and program synthesis. Proceedings of the Merrow Society, 11(4), 349-366.

Varga, D., Haddad, B. (2021). Measuring distributed consensus: a case study. Quarterly Review of Field Studies, 56(1), 171-204.

Varga, R. (1998). A long view of fault tolerance and program synthesis. Journal of Applied Inquiry, 29(4), 302-320.

Whitcombe, C., Haddad, J. (2006). Cache behaviour reconsidered. Quarterly Review of Field Studies, 26(4), 341-357.

Whitcombe, J. (1997). Rethinking program synthesis: evidence from query optimisation. Annals of Regional Research, 31(4), 50-64.

Statement of Purpose

Aiko Whitcombe · Ph.D. in Computer Science, Fall 2027

The department's shared facilities would let me attempt experiments that were out of reach at my previous institution. My interest in query optimisation began during my second year of undergraduate study. I would bring to the program persistence, curiosity and a habit of finishing what I start. That project did not go to plan, and I learned more from the failed first attempt than from the version that worked. Reading the recent literature on query optimisation, I was struck by how many basic questions remain open. Teaching has been a steady part of my education, first as a tutor and later as a laboratory assistant. I am applying to Merrowdale University because its faculty work directly on program synthesis and distributed consensus.

In particular, I hope to study how program synthesis interacts with cache behaviour under realistic conditions. I am comfortable working independently, and I also enjoy the give and take of a research group. I am equally interested in the practical side of query optimisation, which is where I expect my work to be used. After graduating I spent two years in industry, where questions about static analysis came up almost every week. A summer placement gave me my first sustained experience of collecting data on static analysis in the field.

For my senior thesis I designed and ran a small study of query optimisation, which taught me how much care good measurement takes. The seminar sequence in Computer Science would fill the largest gap in my preparation. I believe this background has prepared me well for the demands of the program. I have discussed these ideas with my former advisor, who encouraged me to pursue them at graduate level. In the longer term I hope to lead applied research on cache behaviour, either in a public agency or at a university.

What draws me to this field is that careful work on query optimisation has consequences outside the university. Presenting my results to people outside my own field has been some of the most useful training I have had. My coursework gave me a broad foundation, but it left little time to follow one question in depth. I would welcome the chance to contribute to ongoing projects on query optimisation from my first semester. I want to develop the quantitative skills that my current role only lets me use occasionally.

I came away convinced that the hardest part of research is deciding what not to measure. I have since taught myself the statistical methods that this kind of work depends on. I am applying to Merrowdale University because its faculty work directly on query optimisation and program synthesis. Teaching has been a steady part of my education, first as a tutor and later as a laboratory assistant. I am equally interested in the practical side of fault tolerance, which is where I expect my work to be used.

That project did not go to plan, and I learned more from the failed first attempt than from the version that worked. The seminar sequence in Computer Science would fill the largest gap in my preparation. I want to develop the quantitative skills that my current role only lets me use occasionally. In the longer term I hope to lead applied research on fault tolerance, either in a public agency or at a university. I would bring to the program persistence, curiosity and a habit of finishing what I start.

For my senior thesis I designed and ran a small study of static analysis, which taught me how much care good measurement takes. I came away convinced that the hardest part of research is deciding what not to measure. A summer placement gave me my first sustained experience of collecting data on cache behaviour in the field. I am comfortable working independently, and I also enjoy the give and take of a research group. My coursework gave me a broad foundation, but it left little time to follow one question in depth. After graduating I spent two years in industry, where questions about program synthesis came up almost every week.

My interest in query optimisation began during my second year of undergraduate study. Reading the recent literature on static analysis, I was struck by how many basic questions remain open. I believe this background

has prepared me well for the demands of the program. What draws me to this field is that careful work on cache behaviour has consequences outside the university. I would welcome the chance to contribute to ongoing projects on cache behaviour from my first semester. In particular, I hope to study how static analysis interacts with cache behaviour under realistic conditions. I have since taught myself the statistical methods that this kind of work depends on.

The department's shared facilities would let me attempt experiments that were out of reach at my previous institution. I have discussed these ideas with my former advisor, who encouraged me to pursue them at graduate level. Presenting my results to people outside my own field has been some of the most useful training I have had. That project did not go to plan, and I learned more from the failed first attempt than from the version that worked. In the longer term I hope to lead applied research on fault tolerance, either in a public agency or at a university.

A summer placement gave me my first sustained experience of collecting data on program synthesis in the field. For my senior thesis I designed and ran a small study of query optimisation, which taught me how much care good measurement takes. I am comfortable working independently, and I also enjoy the give and take of a research group. I believe this background has prepared me well for the demands of the program.

Teaching has been a steady part of my education, first as a tutor and later as a laboratory assistant. The seminar sequence in Computer Science would fill the largest gap in my preparation. I have discussed these ideas with my former advisor, who encouraged me to pursue them at graduate level. My coursework gave me a broad foundation, but it left little time to follow one question in depth. I have since taught myself the statistical methods that this kind of work depends on. I am equally interested in the practical side of fault tolerance, which is where I expect my work to be used. I came away convinced that the hardest part of research is deciding what not to measure.

Presenting my results to people outside my own field has been some of the most useful training I have had. I am applying to Merrowdale University because its faculty work directly on cache behaviour and query optimisation. The department's shared facilities would let me attempt experiments that were out of reach at my previous institution. I would bring to the program persistence, curiosity and a habit of finishing what I start. After graduating I spent two years in industry, where questions about fault tolerance came up almost every week. I want to develop the quantitative skills that my current role only lets me use occasionally.

What draws me to this field is that careful work on cache behaviour has consequences outside the university. I would welcome the chance to contribute to ongoing projects on program synthesis from my first semester. Reading the recent literature on program synthesis, I was struck by how many basic questions remain open. My interest in cache behaviour began during my second year of undergraduate study. In particular, I hope to study how program synthesis interacts with cache behaviour under realistic conditions. I am equally interested in the practical side of fault tolerance, which is where I expect my work to be used. Teaching has been a steady part of my education, first as a tutor and later as a laboratory assistant.

GAE — Graduate Aptitude Examination

National Board for Graduate Testing

OFFICIAL SCORE REPORT

Test taker	**Aiko Whitcombe**
Date of birth	**07 Jul 1997**
Registration number	**9256 1251 800**
Test date	**26 Mar 2026**
Report date	**09 Apr 2026**
Intended field	**Computer Science**

Section	Scaled score	Percentile rank
Verbal Reasoning	780	95
Quantitative Reasoning	670	71
Analytical Writing	5.0	91

SCORE RECIPIENTS

Institution	**Merrowdale University (code 7734)**
Department	**Computer Science**

This report is issued to the institution named above at the request of the test taker. It may not be reproduced, and a photocopy is not an official report.

NBGT VERIFIED

Understanding your GAE scores

How scores are reported

Each section is scored separately. Verbal and Quantitative Reasoning are reported on a scale of 200 to 800 in ten-point steps. Analytical Writing is reported on a scale of 0 to 6 in half-point steps.

Percentile ranks

A percentile rank shows the percentage of test takers in the past three years who scored lower. Ranks are updated each July, so the rank for a given score can change from one report to the next.

Comparing scores

Small differences between scores should not be treated as meaningful. A difference of less than thirty points on either reasoning section is within the normal variation between sittings.

Validity

Scores are valid for five years from the test date. Institutions should use scores together with other evidence and never as the only basis for a decision.

Verification

Institutions can confirm any report against the examining board's records using the registration number printed on the front of this report.

Scaled score	Verbal percentile	Quantitative percentile
800	98	92
750	90	84
700	82	76
650	74	68
600	66	60
550	57	51
500	49	43
450	41	35
400	33	27
350	25	19
300	16	10

ROYAL ESTMARK UNIVERSITY

The Trustees, on the recommendation of the Faculty, hereby confer upon

Aiko Whitcombe

the degree of

Bachelor of Science

in Computer Science

with all the rights, honours and privileges pertaining thereto.

Given at Brennhaven, Kingdom of Estmark on 14 Jun 2022

Jonas Okonkwo
President

SEAL OF THE UNIVERSITY

Arjun Nakashima
Registrar

ROYAL ESTMARK UNIVERSITY

Department of Computer Science · Brennhaven, Kingdom of Estmark · n.fairweather@reu.example

04 Nov 2026

Graduate Admissions Committee
Merrowdale University

Re: Recommendation for Aiko Whitcombe, Ph.D. in Computer Science

Dear Members of the Admissions Committee,

I am pleased to write in support of Aiko's application for graduate study. Aiko works well with others and often helped classmates without being asked. The work on fault tolerance has since been taken up by two other members of the group.

When an experiment failed late in the term, Aiko reorganised the work calmly and still met the deadline. I have rarely seen a student absorb criticism so quickly and put it to use. Our weekly meetings were among the most productive I have had with any student.

The final report was clear, careful and honest about its limitations. The quality I value most is a refusal to hide a result that does not fit. In my course on distributed consensus, Aiko earned the highest mark in a class of sixty. Aiko presented this work at our departmental symposium and handled difficult questions with poise.

Colleagues who worked alongside Aiko have told me the same without being prompted. Aiko writes unusually well for someone at this stage. What set Aiko apart was a willingness to question results that looked too tidy.

I have known Aiko for three years, first as a student in two of my courses and later as a research assistant. In discussion Aiko listens carefully and changes position when the evidence calls for it. I would welcome Aiko into my own research group without hesitation. I should add that Aiko did all of this while holding a part-time job. Aiko took responsibility for the data on cache behaviour and kept records that others could follow.

I recommend Aiko to you with enthusiasm and without reservation. Please do not hesitate to contact me if I can provide any further information.

Sincerely,

Naomi Fairweather

Prof. Naomi Fairweather
Associate Professor of Computer Science
Royal Estmark University

TC

Tarnbrook College
Department of Computer Science · Tarnbrook, Ohio · v.hrabal@tarnbrook.example

02 Nov 2026

Graduate Admissions Committee
Merrowdale University

Re: Recommendation for Aiko Whitcombe, Ph.D. in Computer Science

Dear Members of the Admissions Committee,

I am pleased to write in support of Aiko's application for graduate study. Aiko presented this work at our departmental symposium and handled difficult questions with poise. Aiko took responsibility for the data on cache behaviour and kept records that others could follow.

Colleagues who worked alongside Aiko have told me the same without being prompted. Aiko ranks among the top five percent of students I have taught in the past decade. I should add that Aiko did all of this while holding a part-time job. When an experiment failed late in the term, Aiko reorganised the work calmly and still met the deadline.

In discussion Aiko listens carefully and changes position when the evidence calls for it. I expect Aiko to be an independent researcher well before the end of the program. The quality I value most is a refusal to hide a result that does not fit. The final report was clear, careful and honest about its limitations.

I have known Aiko for three years, first as a student in two of my courses and later as a research assistant. Our weekly meetings were among the most productive I have had with any student. Aiko's project on fault tolerance required methods well beyond the syllabus, all of them self-taught.

I would welcome Aiko into my own research group without hesitation. In my course on cache behaviour, Aiko earned the highest mark in a class of sixty. I have rarely seen a student absorb criticism so quickly and put it to use. Aiko writes unusually well for someone at this stage. The work on distributed consensus has since been taken up by two other members of the group.

I recommend Aiko to you with enthusiasm and without reservation. Please do not hesitate to contact me if I can provide any further information.

Sincerely,

Viktor Hrabal

Prof. Viktor Hrabal
Professor of Computer Science
Tarnbrook College

ROYAL ESTMARK UNIVERSITY

Department of Computer Science · Brennhaven, Kingdom of Estmark · h.marchetti@reu.example

19 Oct 2026

Graduate Admissions Committee
Merrowdale University

Re: Recommendation for Aiko Whitcombe, Ph.D. in Computer Science

Dear Members of the Admissions Committee,

I am pleased to write in support of Aiko's application for graduate study. I expect Aiko to be an independent researcher well before the end of the program. Our weekly meetings were among the most productive I have had with any student.

I have known Aiko for three years, first as a student in two of my courses and later as a research assistant. What set Aiko apart was a willingness to question results that looked too tidy. In discussion Aiko listens carefully and changes position when the evidence calls for it.

In my course on program synthesis, Aiko earned the highest mark in a class of sixty. Aiko took responsibility for the data on program synthesis and kept records that others could follow. The quality I value most is a refusal to hide a result that does not fit.

I have rarely seen a student absorb criticism so quickly and put it to use. Aiko presented this work at our departmental symposium and handled difficult questions with poise. I should add that Aiko did all of this while holding a part-time job. I would welcome Aiko into my own research group without hesitation.

Aiko's project on fault tolerance required methods well beyond the syllabus, all of them self-taught. Colleagues who worked alongside Aiko have told me the same without being prompted. When an experiment failed late in the term, Aiko reorganised the work calmly and still met the deadline. Aiko works well with others and often helped classmates without being asked. The final report was clear, careful and honest about its limitations.

I recommend Aiko to you with enthusiasm and without reservation. Please do not hesitate to contact me if I can provide any further information.

Sincerely,

Helena Marchetti

Prof. Helena Marchetti
Assistant Professor of Computer Science
Royal Estmark University