Computer Systems and Professional Practice

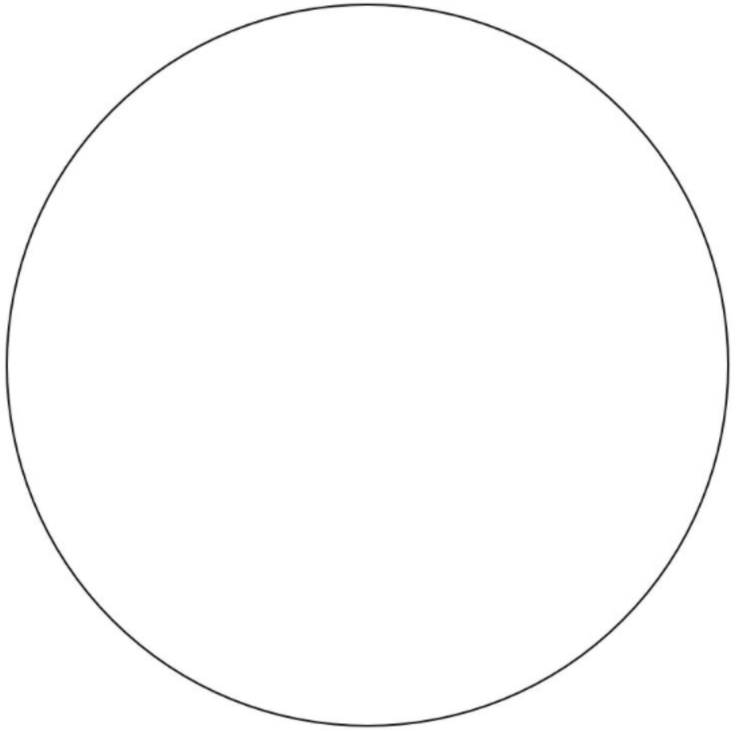
Professor Matthew Leeke
School of Computer Science
University of Birmingham

Academic Writing and Plagiarism

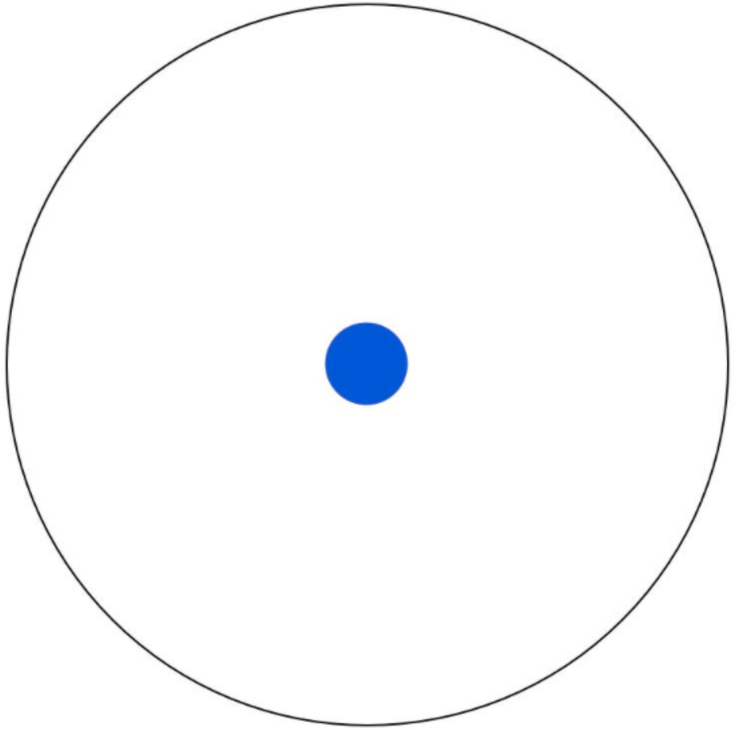
Information Sources

A View of Science

Imagine a circle that contains all of human knowledge:

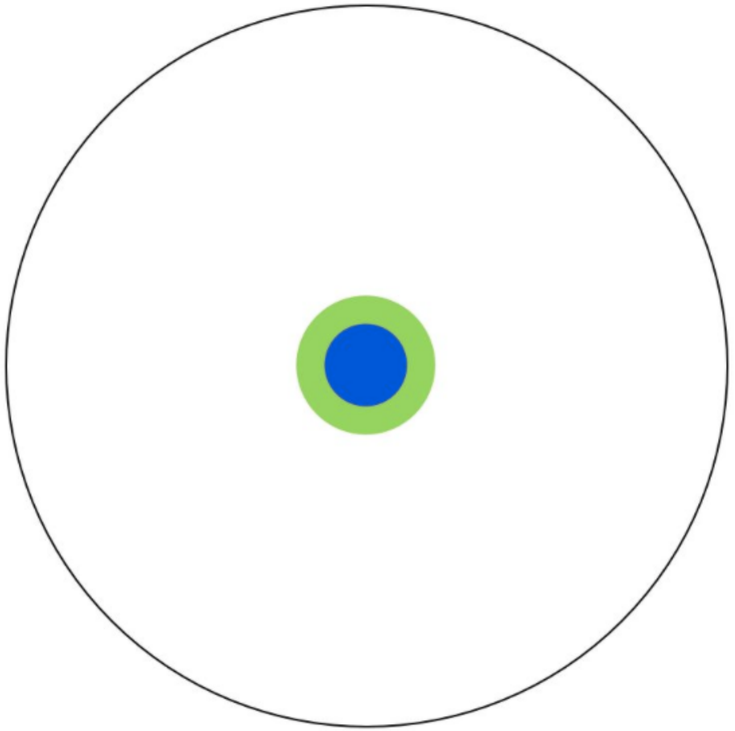


By the time you finish elementary school, you know a little:

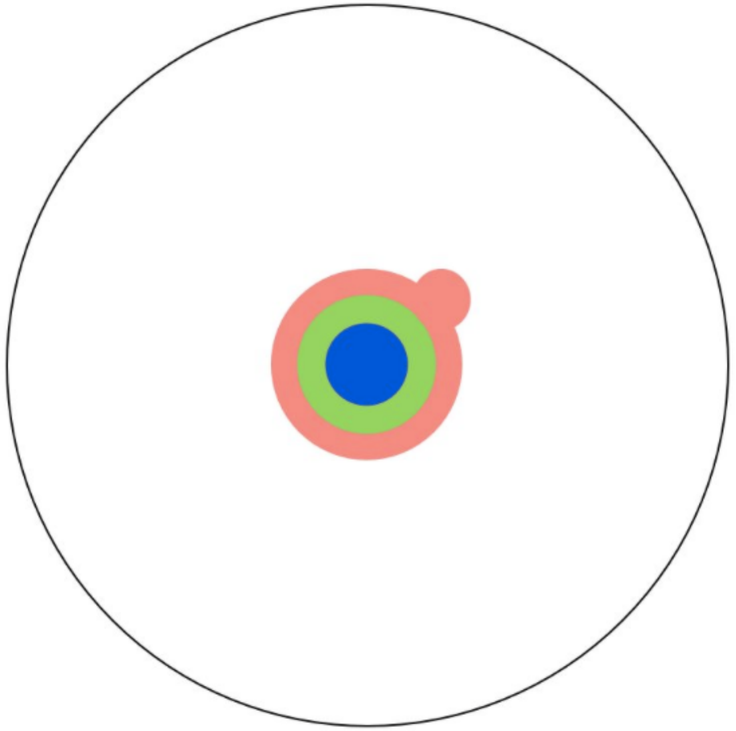


A View of Science

By the time you finish high school, you know a bit more:

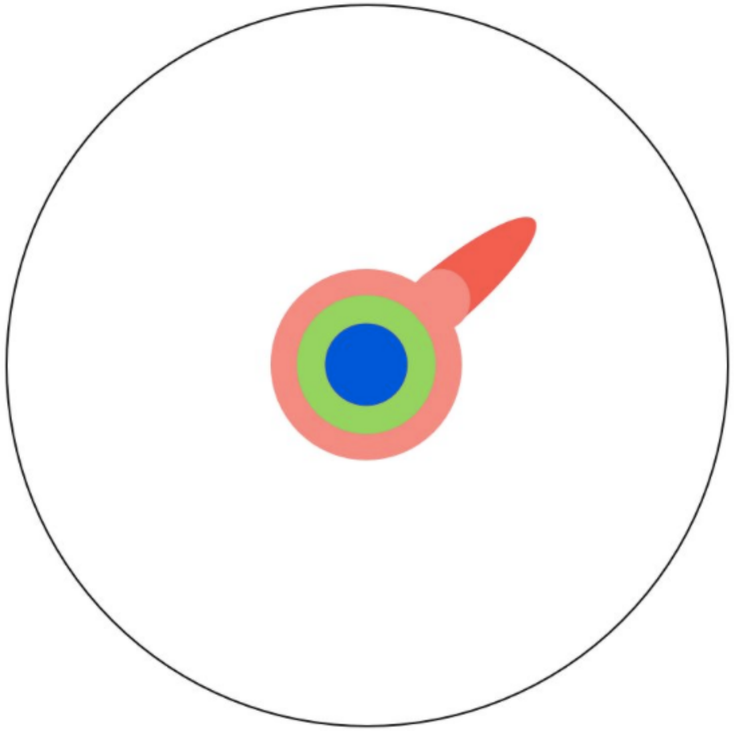


With a bachelor's degree, you gain a specialty:

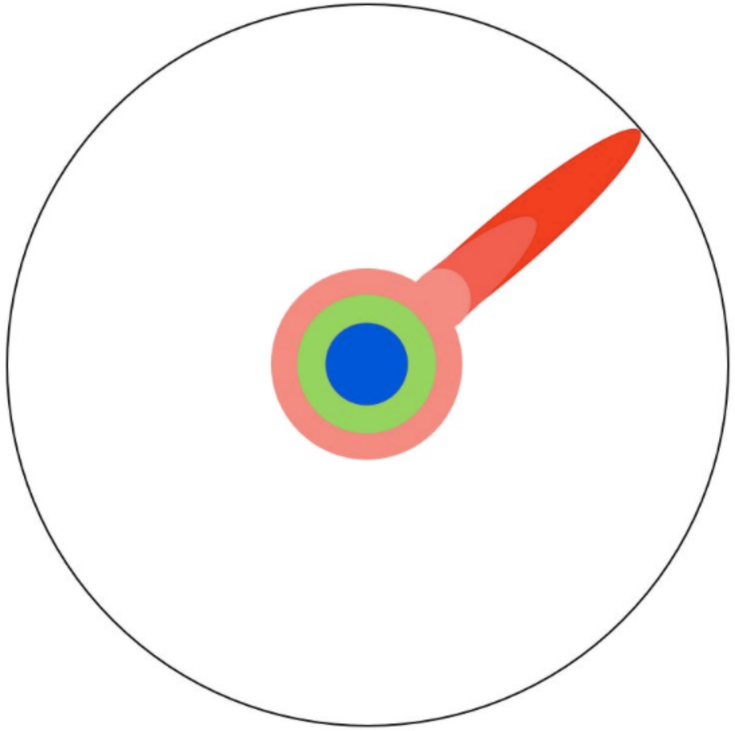


A View of Science

A master's degree deepens that specialty:

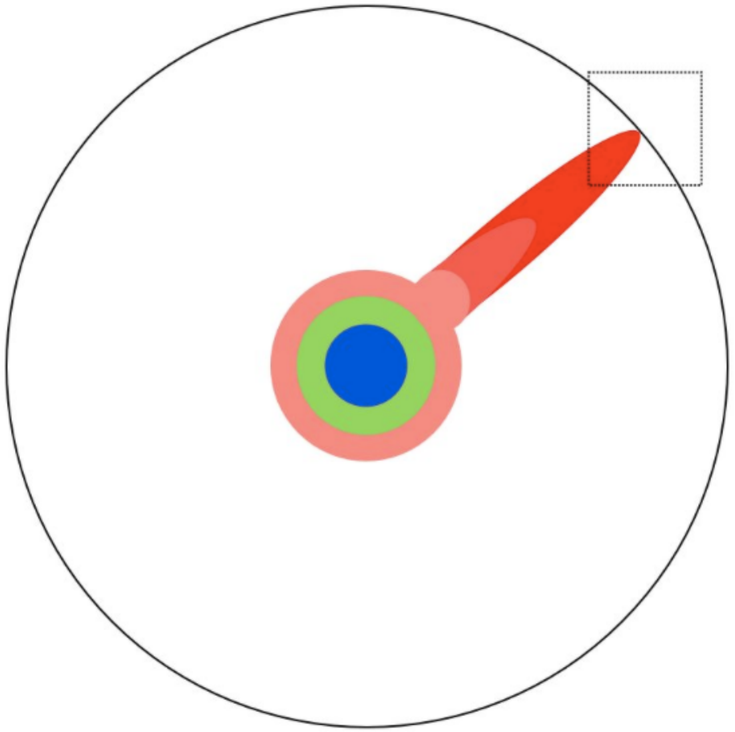


Reading research papers takes you to the edge of human knowledge:

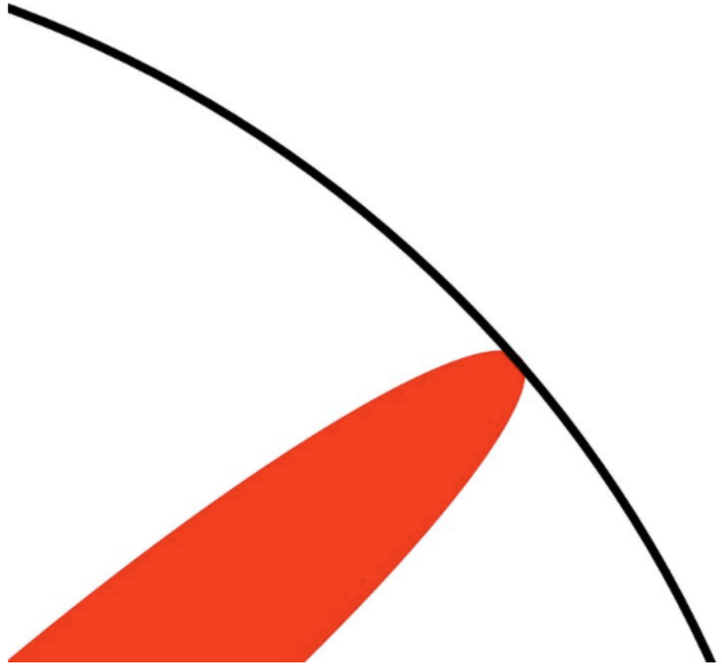


A View of Science

Once you're at the boundary, you focus:

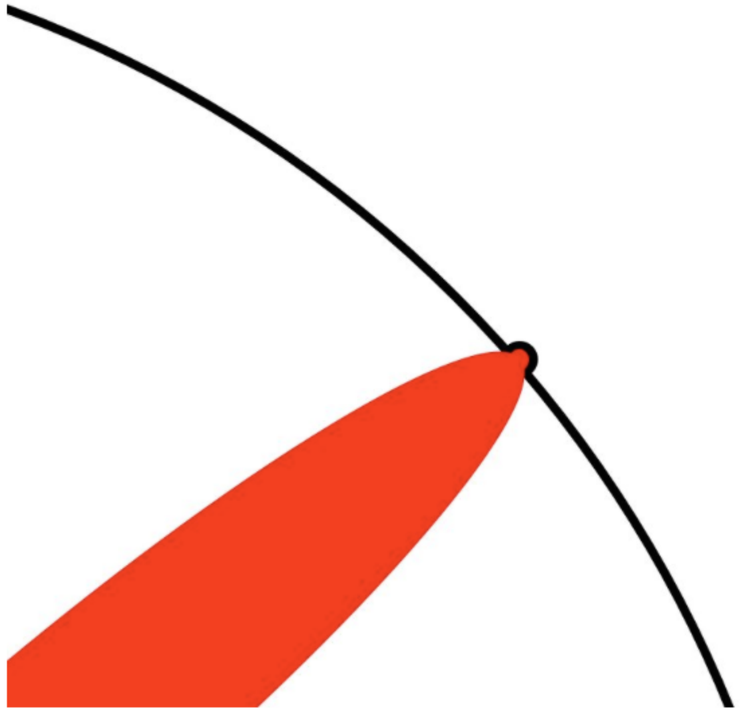


You push at the boundary for a few years:

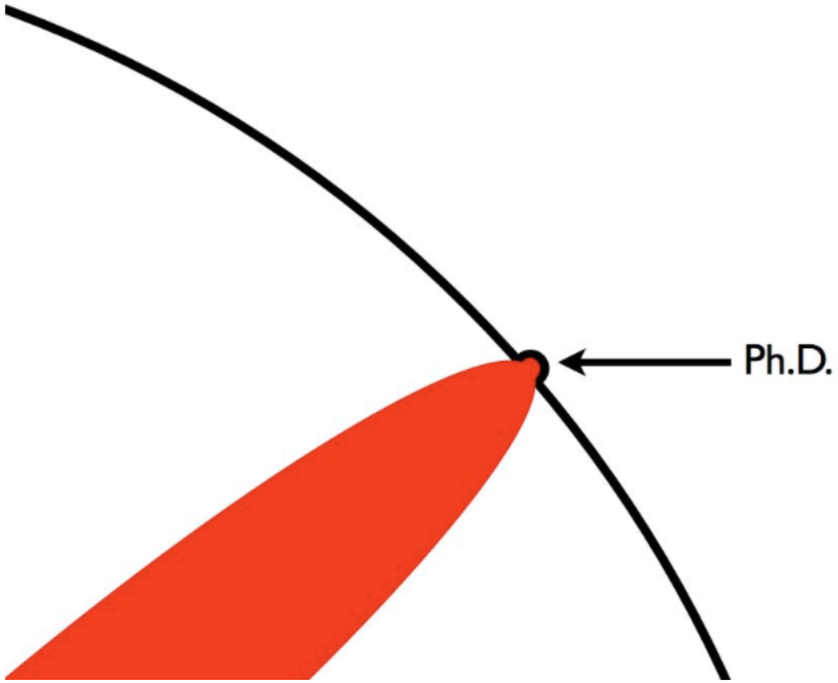


A View of Science

Until one day, the boundary gives way:



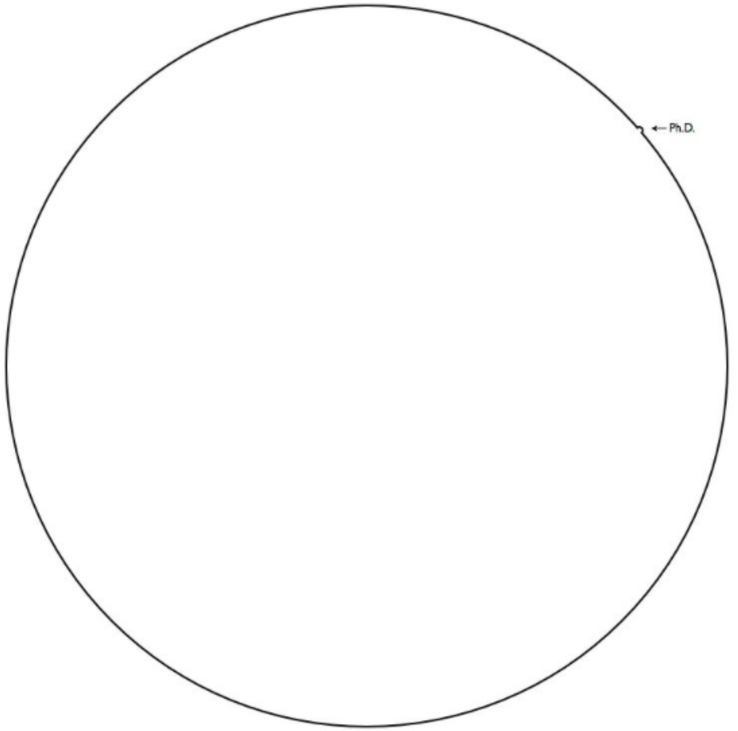
And, that dent you've made is called a Ph.D.:



A View of Science

Of course, the world looks different to you now:

So, don't forget the bigger picture:

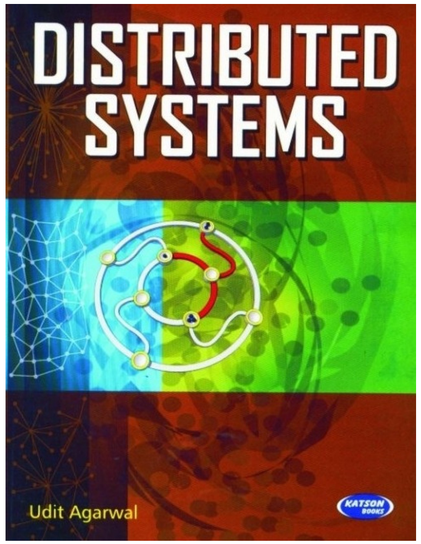


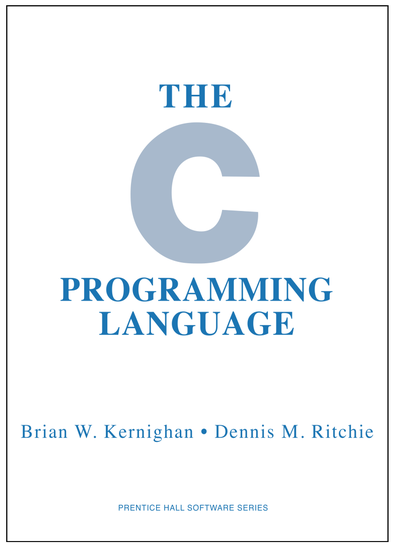
Information Sources

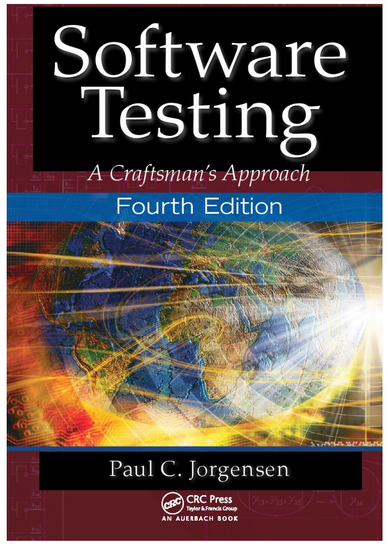
The term “literature” should be understood as content produced by researchers for other researchers

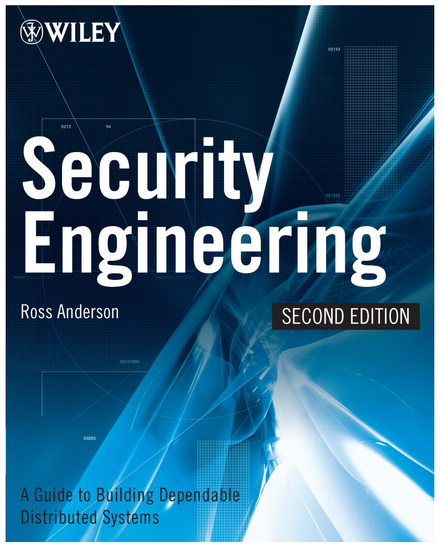
- Books
- Conference and journal papers
- Dissertations, theses and technical reports
- Patents
- Technological standards

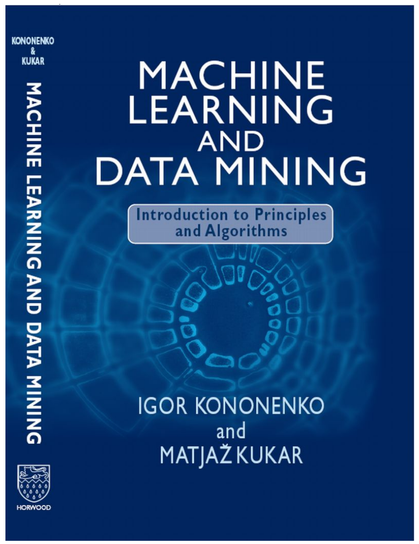
Books


DISTRIBUTED SYSTEMS
Udit Agarwal

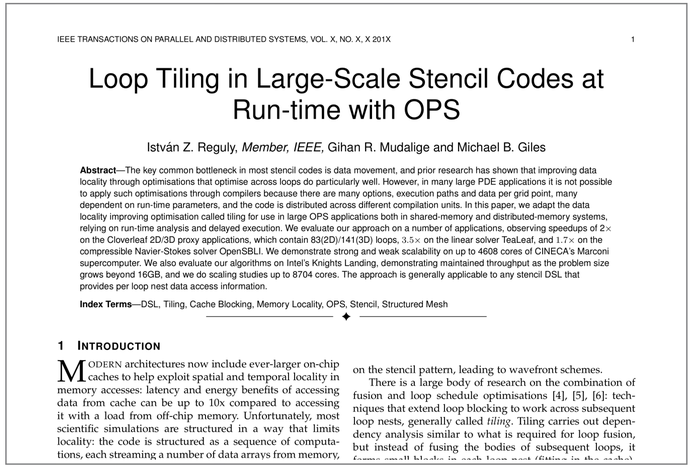

THE
C
PROGRAMMING LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES


Software Testing
A Craftsman's Approach
Fourth Edition
Paul C. Jorgensen
CRC Press
AN AUERBACH BOOK


WILEY
Security Engineering
Ross Anderson
SECOND EDITION
A Guide to Building Dependable Distributed Systems


KONONENKO & KUKAR
MACHINE LEARNING AND DATA MINING
MACHINE LEARNING AND DATA MINING
Introduction to Principles and Algorithms
IGOR KONONENKO and MATJAŽ KUKAR
HORWOOD

Conference and Journal Papers



IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. X, NO. X, X 201X 1

Loop Tiling in Large-Scale Stencil Codes at Run-time with OPS

István Z. Reguly, *Member, IEEE*, Gihan R. Mudalige and Michael B. Giles

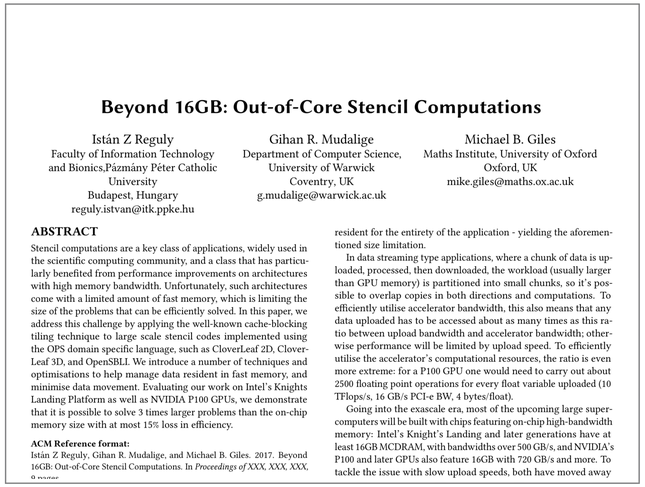
Abstract—The key common bottleneck in most stencil codes is data movement, and prior research has shown that improving data locality through optimisations that optimise across loops do particularly well. However, in many large PDE applications it is not possible to apply such optimisations through compilers because there are many options, execution paths and data per grid point, many dependent on run-time parameters, and the code is distributed across different compilation units. In this paper, we adapt the data locality improving optimisation called tiling for use in large OPS applications both in shared-memory and distributed-memory systems, relying on run-time analysis and delayed execution. We evaluate our approach on a number of applications, observing speedups of 2× on the Cloverleaf 2D/3D proxy applications, which contain 83(2D)/141(3D) loops, 3.5× on the linear solver TeaLeaf, and 1.7× on the compressible Navier-Stokes solver OpenSBLI. We demonstrate strong and weak scalability on up to 4608 cores of CINECA's Marconi supercomputer. We also evaluate our algorithms on Intel's Knights Landing, demonstrating maintained throughput as the problem size grows beyond 16GB, and we do scaling studies up to 8704 cores. The approach is generally applicable to any stencil DSL that provides per loop nest data access information.

Index Terms—DSL, Tiling, Cache Blocking, Memory Locality, OPS, Stencil, Structured Mesh

1 Introduction

MODERN architectures now include ever-larger on-chip caches to help exploit spatial and temporal locality in memory accesses: latency and energy benefits of accessing data from cache can be up to 10x compared to accessing it with a load from off-chip memory. Unfortunately, most scientific simulations are structured in a way that limits locality: the code is structured as a sequence of computations, each streaming a number of data arrays from memory, on the stencil pattern, leading to wavefront schemes.

There is a large body of research on the combination of fusion and loop schedule optimisations [4], [5], [6]: techniques that extend loop blocking to work across subsequent loop nests, generally called *tiling*. Tiling carries out dependency analysis similar to what is required for loop fusion, but instead of fusing the bodies of subsequent loops, it forms small blocks in each loop nest (tiles in the cache)



Beyond 16GB: Out-of-Core Stencil Computations

Istán Z Reguly
Faculty of Information Technology and Bionics,Pázmány Péter Catholic University
Budapest, Hungary
reguly.istvan@itk.ppke.hu

Gihan R. Mudalige
Department of Computer Science, University of Warwick
Coventry, UK
g.mudalige@warwick.ac.uk

Michael B. Giles
Maths Institute, University of Oxford
Oxford, UK
mike.giles@maths.ox.ac.uk

ABSTRACT

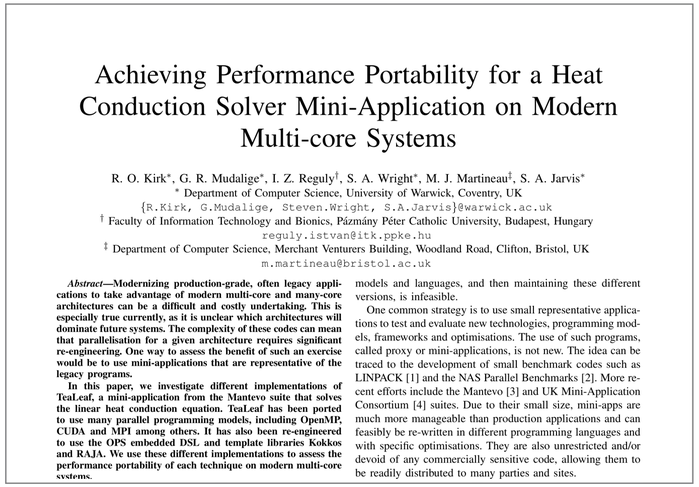
Stencil computations are a key class of applications, widely used in the scientific computing community, and a class that has particularly benefited from performance improvements on architectures with high memory bandwidth. Unfortunately, such architectures come with a limited amount of fast memory, which is limiting the size of the problems that can be efficiently solved. In this paper, we address this challenge by applying the well-known cache-blocking tiling technique to large scale stencil codes implemented using the OPS domain specific language, such as CloverLeaf 2D, CloverLeaf 3D, and OpenSBLI. We introduce a number of techniques and optimisations to help manage data resident in fast memory, and minimise data movement. Evaluating our work on Intel's Knights Landing Platform as well as NVIDIA P100 GPUs, we demonstrate that it is possible to solve 3 times larger problems than the on-chip memory size with at most 15% loss in efficiency.

ACM Reference format:
Istán Z Reguly, Gihan R. Mudalige, and Michael B. Giles. 2017. Beyond 16GB: Out-of-Core Stencil Computations. In *Proceedings of XXX, XXX, XXX,* 9 pages.

resident for the entirety of the application - yielding the aforementioned size limitation.

In data streaming type applications, where a chunk of data is uploaded, processed, then downloaded, the workload (usually larger than GPU memory) is partitioned into small chunks, so it's possible to overlap copies in both directions and computations. To efficiently utilise accelerator bandwidth, this also means that any data uploaded has to be accessed about as many times as this ratio between upload bandwidth and accelerator bandwidth; otherwise performance will be limited by upload speed. To efficiently utilise the accelerator's computational resources, the ratio is even more extreme: for a P100 GPU one would need to carry out about 2500 floating point operations for every float variable uploaded (10 TFlops/s, 16 GB/s PCI-e BW, 4 bytes/float).

Going into the exascale era, most of the upcoming large supercomputers will be built with chips featuring on-chip high-bandwidth memory: Intel's Knight's Landing and later generations have at least 16GB MCDRAM, with bandwidths over 500 GB/s, and NVIDIA's P100 and later GPUs also feature 16GB with 720 GB/s and more. To tackle the issue with slow upload speeds, both have moved away



Achieving Performance Portability for a Heat Conduction Solver Mini-Application on Modern Multi-core Systems

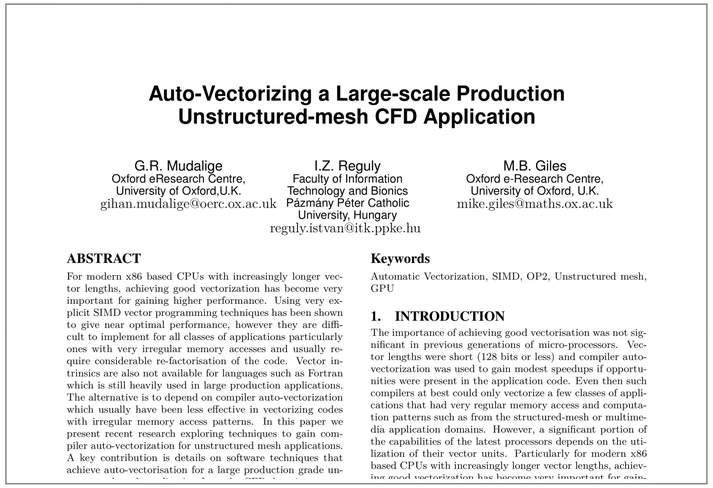
R. O. Kirk*, G. R. Mudalige*, I. Z. Reguly†, S. A. Wright*, M. J. Martineau‡, S. A. Jarvis*
* Department of Computer Science, University of Warwick, Coventry, UK
{R.Kirk, G.Mudalige, Steven.Wright, S.A.Jarvis}@warwick.ac.uk
† Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, Hungary
reguly.istvan@itk.ppke.hu
‡ Department of Computer Science, Merchant Venturers Building, Woodland Road, Clifton, Bristol, UK
m.martineau@bristol.ac.uk

***Abstract*—Modernizing production-grade, often legacy applications to take advantage of modern multi-core and many-core architectures can be a difficult and costly undertaking. This is especially true currently, as it is unclear which architectures will dominate future systems. The complexity of these codes can mean that parallelisation for a given architecture requires significant re-engineering. One way to assess the benefit of such an exercise would be to use mini-applications that are representative of the legacy programs.**

In this paper, we investigate different implementations of TeaLeaf, a mini-application from the Mantevo suite that solves the linear heat conduction equation. TeaLeaf has been ported to use many parallel programming models, including OpenMP, CUDA and MPI among others. It has also been re-engineered to use the OPS embedded DSL and template libraries Kokkos and RAJA. We use these different implementations to assess the performance portability of each technique on modern multi-core systems.

models and languages, and then maintaining these different versions, is infeasible.

One common strategy is to use small representative applications to test and evaluate new technologies, programming models, frameworks and optimisations. The use of such programs, called proxy or mini-applications, is not new. The idea can be traced to the development of small benchmark codes such as LINPACK [1] and the NAS Parallel Benchmarks [2]. More recent efforts include the Mantevo [3] and UK Mini-Application Consortium [4] suites. Due to their small size, mini-apps are much more manageable than production applications and can feasibly be re-written in different programming languages and with specific optimisations. They are also unrestricted and/or devoid of any commercially sensitive code, allowing them to be readily distributed to many parties and sites.



Auto-Vectorizing a Large-scale Production Unstructured-mesh CFD Application

G.R. Mudalige
Oxford eResearch Centre, University of Oxford,U.K.
gihan.mudalige@oerc.ox.ac.uk

I.Z. Reguly
Faculty of Information Technology and Bionics
Pázmány Péter Catholic University, Hungary
reguly.istvan@itk.ppke.hu

M.B. Giles
Oxford e-Research Centre, University of Oxford, U.K.
mike.giles@maths.ox.ac.uk

ABSTRACT

For modern x86 based CPUs with increasingly longer vector lengths, achieving good vectorization has become very important for gaining higher performance. Using very explicit SIMD vector programming techniques has been shown to give near optimal performance, however they are difficult to implement for all classes of applications particularly ones with very irregular memory accesses and usually require considerable re-factorisation of the code. Vector intrinsics are also not available for languages such as Fortran which is still heavily used in large production applications. The alternative is to depend on compiler auto-vectorization which usually have been less effective in vectorizing codes with irregular memory access patterns. In this paper we present recent research exploring techniques to gain compiler auto-vectorization for unstructured mesh applications. A key contribution is details on software techniques that achieve auto-vectorisation for a large production grade un-

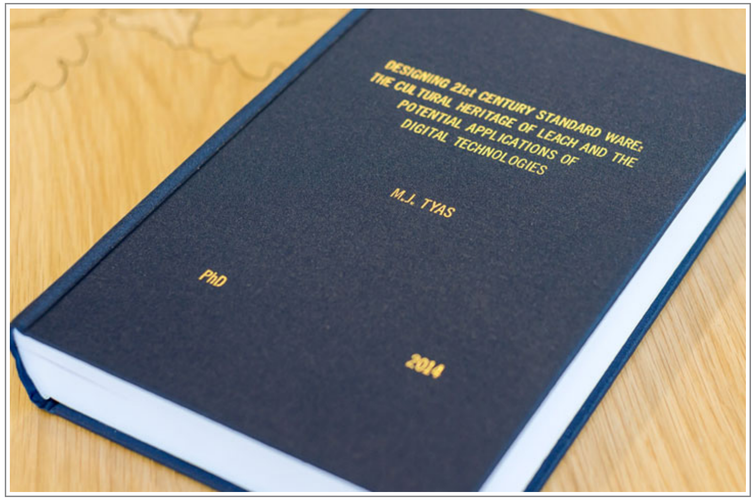
Keywords

Automatic Vectorization, SIMD, OP2, Unstructured mesh, GPU

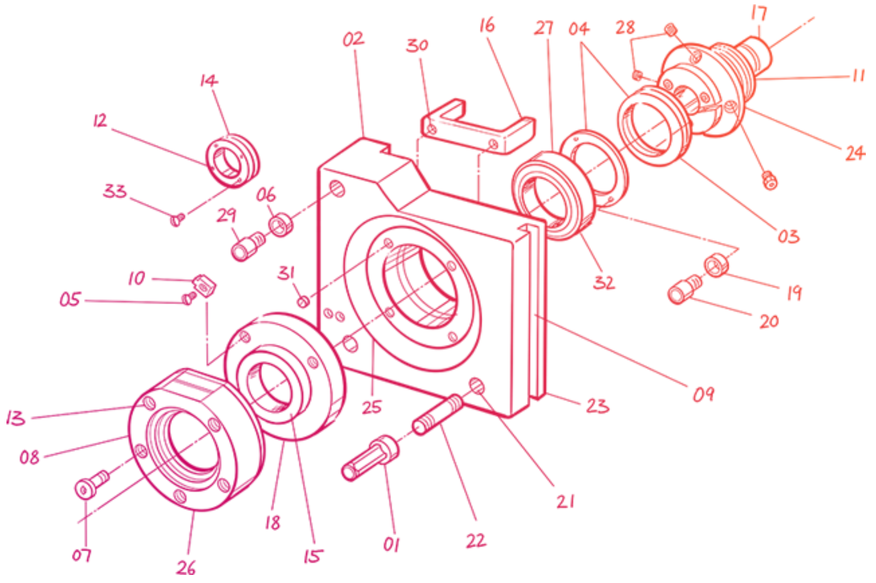
1. INTRODUCTION

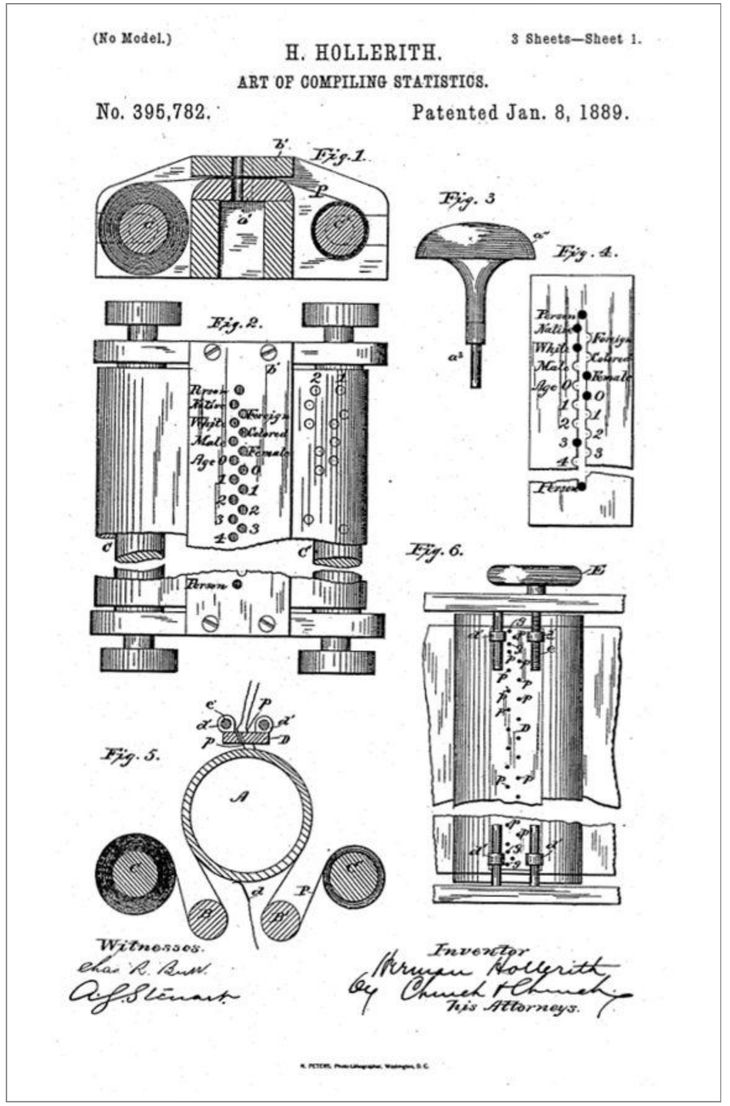
The importance of achieving good vectorisation was not significant in previous generations of micro-processors. Vector lengths were short (128 bits or less) and compiler auto-vectorization was used to gain modest speedups if opportunities were present in the application code. Even then such compilers at best could only vectorize a few classes of applications that had very regular memory access and computation patterns such as from the structured-mesh or multimedia application domains. However, a significant portion of the capabilities of the latest processors depends on the utilization of their vector units. Particularly for modern x86 based CPUs with increasingly longer vector lengths, achiev-

Dissertations, Theses and Technical Reports



Patents




(No Model.) 3 Sheets—Sheet 1.

H. HOLLERITH.

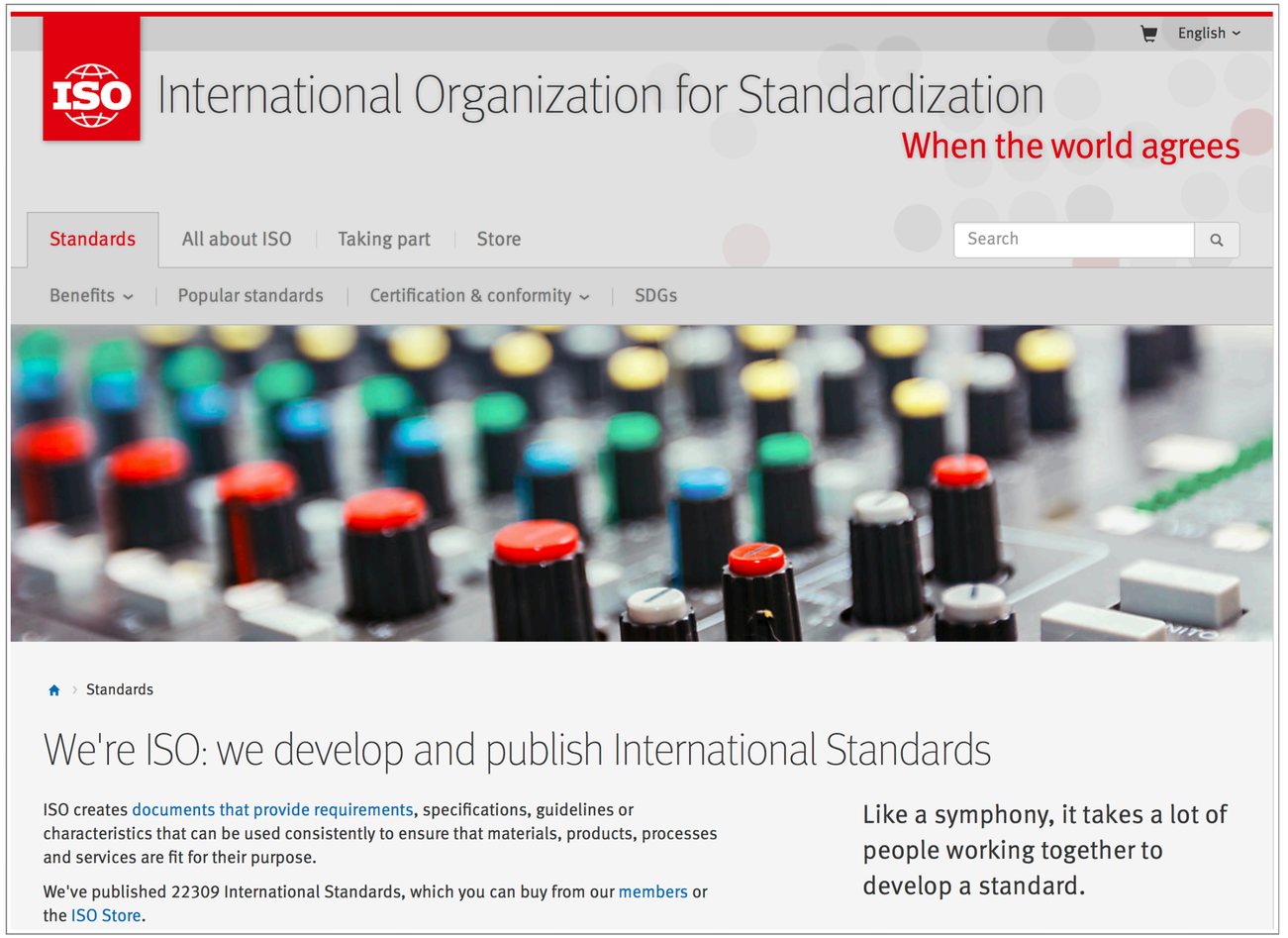
ART OF COMPILING STATISTICS.

No. 395,782. Patented Jan. 8, 1889.

Witnesses.
Chas. R. Burr
A. J. Stinnart

Inventor
Herman Hollerith
by Church & Church
his Attorneys.

Technological Standards



The Importance of Peer Review

Peer reviews subjects an author's scholarly work, research or ideas to the scrutiny of others who are experts in the same field

It has many benefits but two primary purposes

1. A filter to ensure that only high quality research is published, especially in reputable journals, by determining validity, significance and originality

2. A mechanisms to improve the quality of manuscripts that are deemed suitable for publication

Peer review is at the heart of scientific research

Reputable Sources

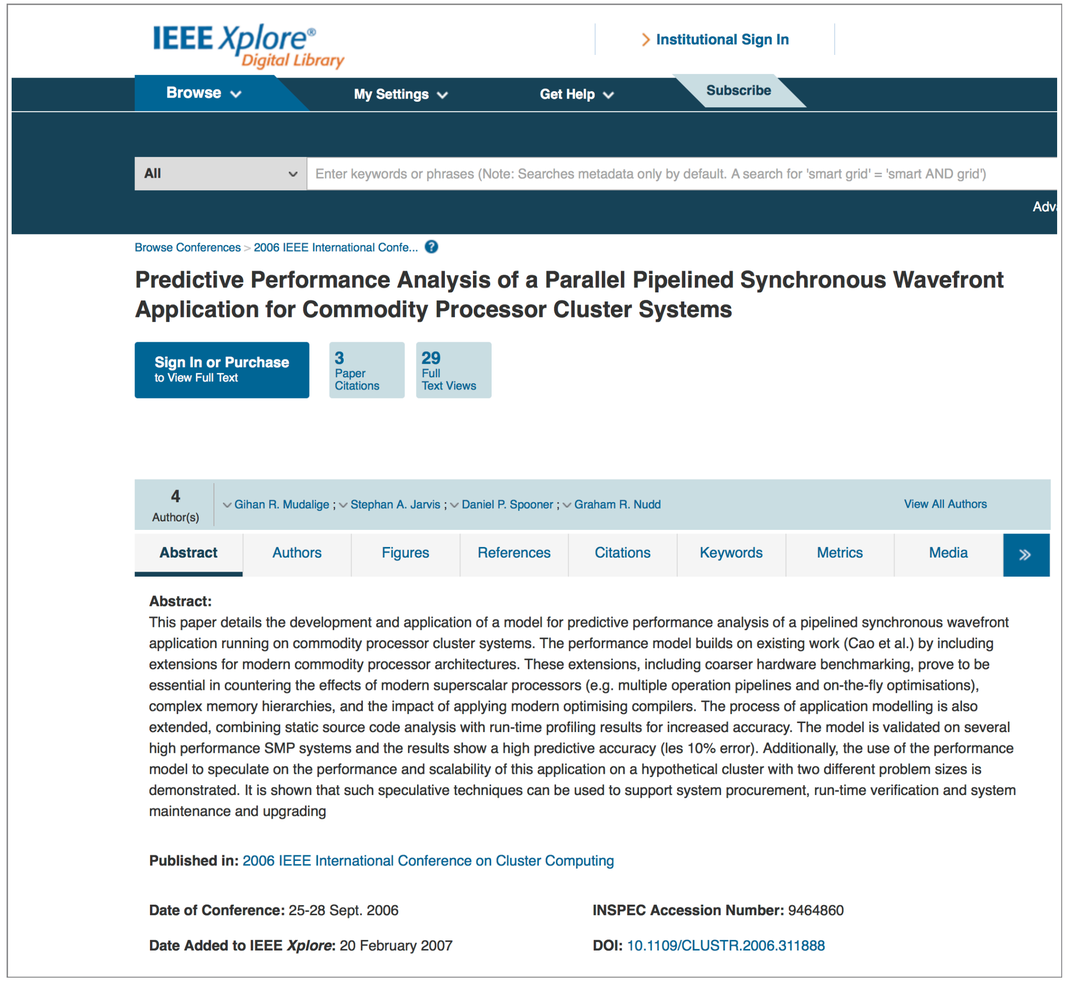
Publication in peer-reviewed conferences and journals ensures a certain level of confidence in research quality

Magazines, websites, blogs, news articles and similar sources can be useful in scientific research but their role is not to form the basis of a sound argument

It may alert the reader to the existence of reputable work, but is rarely worth citing

Learning might be built on a wider literature, but the arguments in the write-up should be based on knowledge from refereed sources

Reputable Sources?



IEEE Xplore® Digital Library

Institutional Sign In

Browse | My Settings | Get Help | Subscribe

All | Enter keywords or phrases (Note: Searches metadata only by default. A search for 'smart grid' = 'smart AND grid')

Browse Conferences > 2006 IEEE International Confe...

Predictive Performance Analysis of a Parallel Pipelined Synchronous Wavefront Application for Commodity Processor Cluster Systems

Sign In or Purchase to View Full Text

3 Paper Citations

29 Full Text Views

4 Author(s) — Gihan R. Mudalige ; Stephan A. Jarvis ; Daniel P. Spooner ; Graham R. Nudd — View All Authors

Abstract | Authors | Figures | References | Citations | Keywords | Metrics | Media

Abstract:
This paper details the development and application of a model for predictive performance analysis of a pipelined synchronous wavefront application running on commodity processor cluster systems. The performance model builds on existing work (Cao et al.) by including extensions for modern commodity processor architectures. These extensions, including coarser hardware benchmarking, prove to be essential in countering the effects of modern superscalar processors (e.g. multiple operation pipelines and on-the-fly optimisations), complex memory hierarchies, and the impact of applying modern optimising compilers. The process of application modelling is also extended, combining static source code analysis with run-time profiling results for increased accuracy. The model is validated on several high performance SMP systems and the results show a high predictive accuracy (les 10% error). Additionally, the use of the performance model to speculate on the performance and scalability of this application on a hypothetical cluster with two different problem sizes is demonstrated. It is shown that such speculative techniques can be used to support system procurement, run-time verification and system maintenance and upgrading

Published in: 2006 IEEE International Conference on Cluster Computing

Date of Conference: 25-28 Sept. 2006

INSPEC Accession Number: 9464860

Date Added to IEEE *Xplore*: 20 February 2007

DOI: 10.1109/CLUSTR.2006.311888

WIKIPEDIA
The Free Encyclopedia

Article | Talk | Read

Wiki Loves Monuments: Photograph a monument, help Wikipec

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
বাংলা
Deutsch
Ελληνικά
Français
Italiano
മലയാളം
ਪੰਜਾਬੀ
Scots
中文

Supercomputer

From Wikipedia, the free encyclopedia
(Redirected from High performance computing)

"High-performance computing" redirects here. For narrower definitions of HPC, see High-throughput computing and Many-task computi

A **supercomputer** is a computer with a high level of performance compared to a general-purpose computer. Performance of a supercompu floating-point operations per second (FLOPS) instead of million instructions per second (MIPS). As of 2017, there are supercomputers whic nearly a hundred quadrillion FLOPS.[3] As of November 2017, all of the world's fastest 500 supercomputers run Linux-based operating syst research is being conducted in China, the United States, the European Union, Taiwan and Japan to build even faster, more powerful and m superior exascale supercomputers.[5]

Supercomputers play an important role in the field of computational science, and are used for a wide range of computationally intensive tas including quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling (computing the structures chemical compounds, biological macromolecules, polymers, and crystals), and physical simulations (such as simulations of the early mome airplane and spacecraft aerodynamics, the detonation of nuclear weapons, and nuclear fusion). Throughout their history, they have been es cryptanalysis.[6]

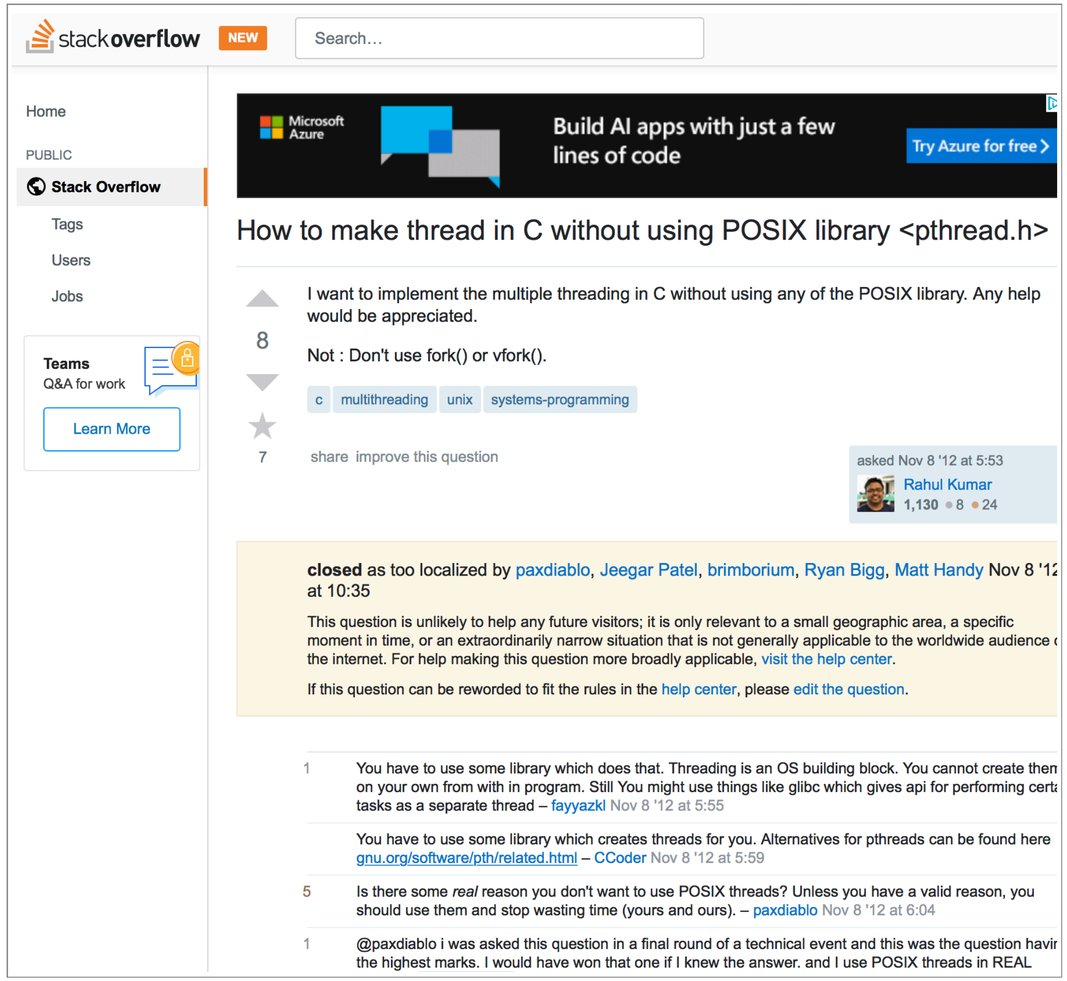
Supercomputers were introduced in the 1960s, and for several decades the fastest were made by Seymour Cray at Control Data Corporati Research and subsequent companies bearing his name or monogram. The first such machines were highly tuned conventional designs tha their more general-purpose contemporaries. Through the 1960s, they began to add increasing amounts of parallelism with one to four proc From the 1970s, the vector computing concept with specialized math units operating on large arrays of data came to dominate. A notable e successful Cray-1 of 1976. Vector computers remained the dominant design into the 1990s. From then until today, massively parallel super of thousands of off-the-shelf processors became the norm.[7][8]

The US has long been a leader in the supercomputer field, first through Cray's almost uninterrupted dominance of the field, and later throug field in the 1980s and 90s, but since then China has become increasingly active in the field. As of June 2018, the fastest supercomputer on a LINPACK benchmark score of 122.3 PFLOPS, exceeding the previous record holder, Sunway TaihuLight, by around 29 PFLOPS.[3][9] Sur Chinese computer to enter the TOP500 list without using hardware from the United States. As of June 2018, China had more computers (2 computers held eight of the top 20 positions;[10][11] the U.S. has six of the top 10 and China has two.

Contents [hide]

1 History
  1.1 Massively parallel designs
2 Special purpose supercomputers
3 Energy usage and heat management
4 Software and system management
  4.1 Operating systems
  4.2 Software tools and message passing
5 Distributed supercomputing

Reputable Sources?



Digital Libraries and Search Engines

Many sources for published research in Computer Science

ACM Digital Library, Google Scholar, IEEE Xplore Digital Library etc..

Important to carefully consider the origins of every resource used

Advice on Information Sources

Peer reviewed scientific papers, whether from a conference or journal, are the gold standard for reputability

News articles, patents, technical standards, etc. can be useful when framing work or making a particular point

Learning to reference is absolutely essential

Plagiarism

What Is Plagiarism?

Plagiarism is re-use of others' ideas, words, material without appropriate acknowledgement of the sources of that information

This includes, not just copying from published papers, but from material in electronic form, such as websites, news articles, etc.

What Is Appropriate Acknowledgement?

A common way that students commit plagiarism is by using phrases taken from a published source without proper acknowledgment.

Proper acknowledgement has two requirements:

1. An indication of the source of the quotation

2. An indication of the extent of the quotation

Case Study 1

ORIGINAL TEXT:

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface.

[1] B. Tognazzini, First Principles of Interaction Design, 2003, [Online] http://www.asktog.com/basics/firstprinciples.html [Accessed 14th September 2018]

YOUR TEXT:

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface [1].

Case Study 1 - It’s plagiarism!

ORIGINAL TEXT:

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface.

[1] B. Tognazzini, First Principles of Interaction Design, 2003, [Online] http://www.asktog.com/basics/firstprinciples.html [Accessed 14th September 2018]

YOUR TEXT:

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface [1].

Case Study 1 - Why?

Simply placing the source for your quote into your reference list is not sufficient

Doesn’t meet the second requirement (an indication of the extent of the quotation)

The reader cannot judge exactly what has been drawn from the quoted source and what has not.

Case Study 1 - Use Quotations

Use it for a group of words taken from a text or speech.

When you quote, place the words you are using in quotation marks, and document the relevant source

Case Study 1 - Corrected

ORIGINAL TEXT:

The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface.

[1] B. Tognazzini, First Principles of Interaction Design, 2003, [Online] http://www.asktog.com/basics/firstprinciples.html [Accessed 14th September 2018]

YOUR TEXT:

"The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface" [1].

Guidelines for Using Quotations

Use direct quotations only when the author's wording is necessary or particularly effective

Don’t use excessive quotations in your paper

Avoid quoting more than is needed

A quotation should add weight to what you are trying to say, not repeat a point you have made

Guidelines for Using Quotations

Be sure to integrate all quotations into your own discussion

Introduce direct quotations with your own words

YOUR TEXT WITH OWN WORDS INTRODUCTION:

Since 2003 it has been argued that “The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface” [1].

Case Study 2

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-driven companies became more visible in the eastern part of the country, they changed farm hands into factory workers, and provided jobs for the large wave of immigrants. Therefore, the increase of industry, the growth of cities, and the explosion of the population were three large factors of nineteenth century America.

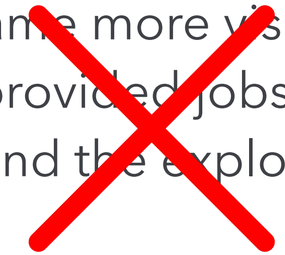
Case Study 2 - It’s plagiarism!

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-driven companies became more visible in the eastern part of the country, they changed farm hands into factory workers, and provided jobs for the large wave of immigrants. Therefore, the increase of industry, the growth of cities, and the explosion of the population were three large factors of nineteenth century America.



Case Study 2 - Why?

The writer has failed to cite a source for any of the ideas or facts

Easily detected by any informed reader, particularly if they’re versed in the field of study

May have done deliberately or inadvertently

Case Study 3

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-driven companies became more visible in the eastern part of the country, they changed farm hands into factory workers, and provided jobs for the large wave of immigrants. Therefore, the increase of industry, the growth of cities, and the explosion of the population were three large factors of nineteenth century America [2].

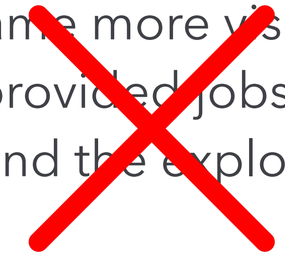
Case Study 3 - It’s plagiarism!

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-driven companies became more visible in the eastern part of the country, they changed farm hands into factory workers, and provided jobs for the large wave of immigrants. Therefore, the increase of industry, the growth of cities, and the explosion of the population were three large factors of nineteenth century America [2].



Case Study 3 - Why?

Although the writer has cited a source, they have only changed around a few words and phrases and/or changed the order of the original sentences

Paraphrasing

Use someone’s ideas, but put them in your own words, and acknowledge the source of the information

Widely used where writers are versed in a field of study

Case Study 4

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

Steam-powered production had shifted labor from agriculture to manufacturing, and as immigrants arrived in the US, they found work in these new factories. As a result, populations grew, and large urban areas arose. This is how America developed in the late nineteenth century.

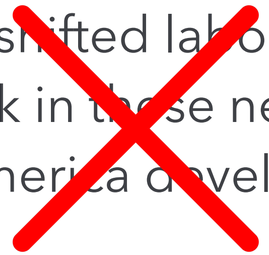
Case Study 4 - It’s plagiarism!

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

Steam-powered production had shifted labor from agriculture to manufacturing, and as immigrants arrived in the US, they found work in these new factories. As a result, populations grew, and large urban areas arose. This is how America developed in the late nineteenth century.



Case Study 4 - Why?

Although the writer has paraphrased the text, they have failed to cite the source

Case Study 5

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

Steam-powered production had shifted labor from agriculture to manufacturing, and as immigrants arrived in the US, they found work in these new factories. As a result, populations grew, and large urban areas arose. This is how America developed in the late nineteenth century [2].

Case Study 5 - It’s fine!

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

Steam-powered production had shifted labor from agriculture to manufacturing, and as immigrants arrived in the US, they found work in these new factories. As a result, populations grew, and large urban areas arose. This is how America developed in the late nineteenth century [2].

Case Study 5 - Why?

Paragraph relays the information in the original text using the own words

The source of the information is cited

Extent is clear from citation

Case Study 6

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-powered production shifted labor from agriculture to manufacturing, the demand for workers "transformed farm hands into industrial labourers," and created jobs for immigrants. In turn, growing populations increased the size of urban areas [2].

Case Study 6 - It’s fine!

ORIGINAL TEXT:

The rise of industry, the growth of cities, and the expansion of the population were the three great developments of late nineteenth century American history. As new, larger, steam-powered factories became a feature of the American landscape in the East, they transformed farm hands into industrial labourers, and provided jobs for a rising tide of immigrants.

YOUR TEXT:

As steam-powered production shifted labor from agriculture to manufacturing, the demand for workers "transformed farm hands into industrial labourers," and created jobs for immigrants. In turn, growing populations increased the size of urban areas [2].

Case Study 6 - Why?

Mixing paraphrasing and quotation where the source of the information describes in the sentence is cited

The source of the information is cited

Extent is clear from citation

Common Knowledge

No need to cite the source when the information is the **common knowledge**, i.e., the facts that are common sense, can be found in numerous places, and are likely to be known by a lot of people.

For example:

A parallel application can be run on multiple processors simultaneously

The binary search algorithm has a logarithmic complexity

Figures, Tables and Data

If you are using the figures and tables coming from the sources, you MUST cite the source in the figure caption or table title

For example:

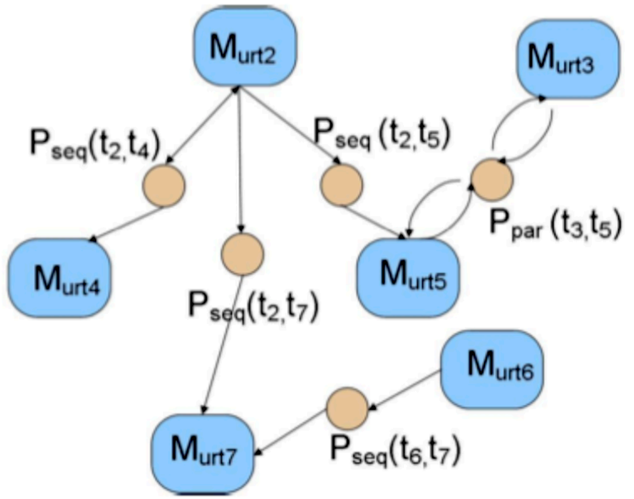
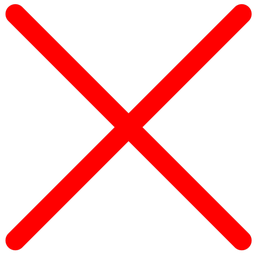


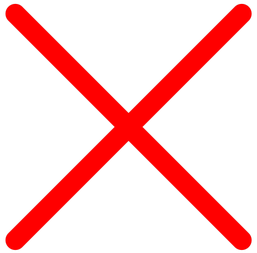
Figure 1. Hierarchy of the authorization control module for the workflow [1]

Plagiarism Summary

Copy without reference



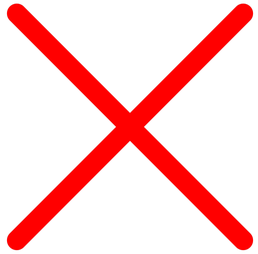
Copy with reference



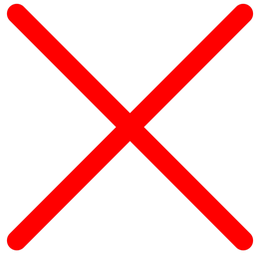
Change a few words or order without reference

Plagiarism Summary

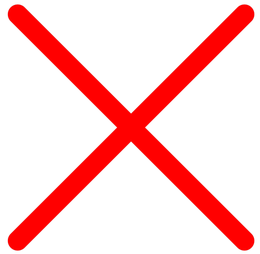
Change a few words or order with reference



Use figures and tables without reference



Paraphrase without reference



Plagiarism Summary

Quotation with reference 

Paraphrase with reference 

Use figure, table or data with reference 

Avoiding Plagiarism

Put in quotations everything that comes directly from the text in the source, trying to provide some introduction with your own words

Paraphrase but be sure you are not just rearranging or replacing a few words

A good approach is to:

1. Read the original text

2. Close the text

3. Write out the idea in your own words

Assisting Plagiarism is Plagiarism

Assisting plagiarism is considered as serious as conducting plagiarism

To protect yourself, do not lend any part of your solution to any student until after the assignment has been marked and returned

Keep all of your old notes and drafts until after the end of the course (so that if something does happen, you have proof that you did the work yourself).

Detecting Plagiarism

Detecting plagiarism

All assignments are submitted to plagiarism detctions systems, including TurnItIn

Compares your work against published articles, web pages, and student papers to generate a report

14 billion web pages

Leading libraries and academic databases

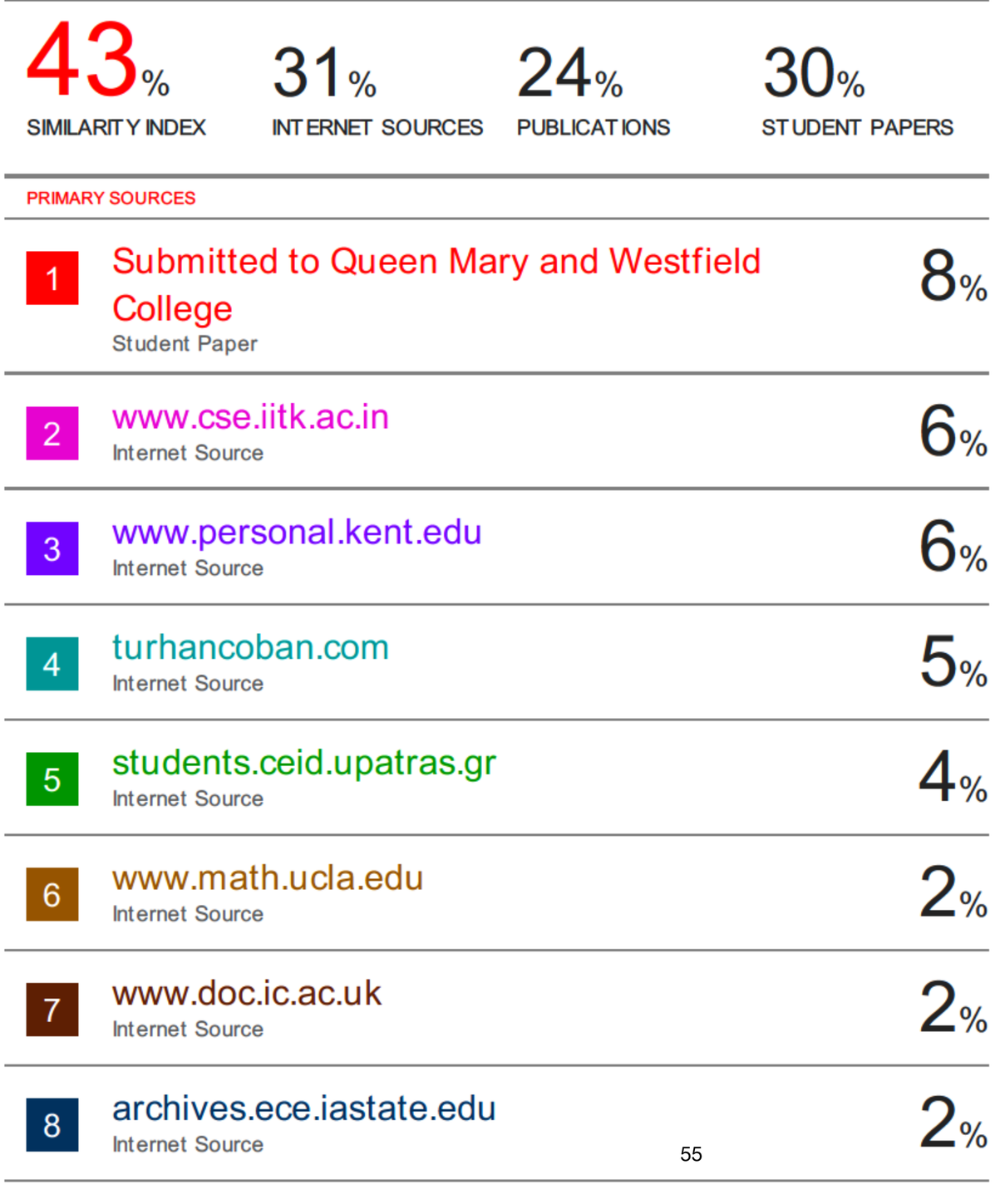
Student repositories including 150 million student papers

Detecting Plagiarism

Checks for similarity with coursework submissions at many institutions, online sources, academic repositories, etc..

Not just a bag of words - structural and thematic similarities

Returns quantified analysis for interpretation



43%	31%	24%	30%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Queen Mary and Westfield College Student Paper	8%
2	www.cse.iitk.ac.in Internet Source	6%
3	www.personal.kent.edu Internet Source	6%
4	turhancoban.com Internet Source	5%
5	students.ceid.upatras.gr Internet Source	4%
6	www.math.ucla.edu Internet Source	2%
7	www.doc.ic.ac.uk Internet Source	2%
8	archives.ece.iastate.edu Internet Source	2%

Advice on Plagiarism

No need to worry if you learn how to reference your sources and do not attempt to claim the work of others as your own

Communicate using your own words and understanding, whilst citing relevant work

If you are in doubt, speak to your supervisor

Academic Writing

Writing Coursework and Academic Papers

What is the purpose of a coursework report?

What do your readers want to find out?

How will they be reading your report?

Is there a difference between a coursework report and an academic paper?

Characterising Academic Writing

Communicates with its reader clearly and persuasively

High quality content

Logical structure

Precise style

Conventions to show how source material has been used

Language

Avoid exaggeration, colloquialisms and anecdotal language

> It’s nice that you think your idea is “excellent” but your reviewer may not agree

Avoid contradiction, cliches and abbreviations

> At the end of the day, it’s like chalk and cheese

Be specific and assertive where possible, avoiding words like ‘may’, ‘could’, ‘perhaps’, ‘might’, etc. unless you intend to be speculative

> Precision and restraint are important when expressing yourself

Can You Be Precise?

Can you shorten the sentence below to no more than 17 words?

"There has been much less research in education during the past year due to the fact that there is a complete lack of public funding for it." (27 words)

Can I Be Precise?

“There has been much less research in education during the past year due to the fact that there is a complete lack of public funding for it.” (27 words)

“A lack of public funding has led to less research in education this year.” (14 words)

Let’s Be Practical

Can we characterise what a general report might look like for a software engineering project?

- Documenting a development process
- Second year project
- Final year project

Let’s try it…

Report Structure - An Example for Software Projects

Project Report - How Should I Approach Writing?

Plan the structure of your report

Titles and subtitles

Can you write a comment on what goes in each subsection?

Do these comments tell the story that you want?

Draft. Read. Redraft. Reread. Redraft. Reread. Redraft. Reread…

A Report Structure for Software Projects

Abstract - Usually 100-300 words stating the salient points of the report. It should help your reader to decide whether the report is relevant to her or his interests.

Introduction - The introduction should provide content for the report, discuss relevant background material, identify stakeholders, and state the aim(s) of the work.

Research - This is often referred to the ‘literature review’ section. It is one of the most important section of the Project Proposal and the Final Report. It is where you demonstrate that you understand the state-of-the-art in the field you’re working. Towards the end of this sections it’s normally a good idea to explain how your aim / work / idea / contribution differs from the nearest work in the field.

A Report Structure for Software Projects

Legal, Social, Ethical and Professional Issues - Address the legal, social, ethical, and professional issues associated with your project. Find something to say for each. For example, listening for legal, BCS Code of Conduct for professional, etc..

System Requirements - This section should detail your understanding of what you are planning to create. The section should aim to break down the overarching aims of the work into clear, measurable requirements that can be used in the evaluation of the project. This is why we often function of functional and non-functional requirements.

Design - Communicating how you think about the composition of your system and how it works. You might detail the ways in which the overall system will be broken down into subsystems. Detail should then be provided on the design of each of these subsystems.

A Report Structure for Software Projects

Implementation - This section should discuss how you went about developing a system that was consistent with your design to meet your stated requirements. The implementation of subsystems should be accurately documented, with any implementation difficulties being acknowledged. The Design and Implementation sections can be grouped in the Final Report, if these are tightly coupled. Likely omitted for the Project Proposal, though should mention your proposed implementation technologies somewhere.

Testing and Success Measurement - As well as documenting system testing, this section should also describe any unit testing or integration testing performed. If you are not familiar with unit, integration or system testing then it would be a good idea to investigate these notions and consider about how they relate to your project. This section might also detail any performance, reliability or usability testing performed, with quantification, i.e., numeric measurements, being used wherever possible. All those points are systems focused through. If you're doing something that's research focused or more of a social / analytical study then you need to think about how you'll measure success.

A Report Structure for Software Projects

Project Management - What is the timeline for your project? What software development methodology will you use? Can you identify the major milestones? These types of questions are important for project planning and should be addressed directly.

Evaluation - Usually you evaluate your project with regard to the functional and non-functional requirements you set out in the earlier chapter. This doesn't necessary mean that your project was successful but, if these requirements were appropriately specified, you it's likely that your project was successful. You might be reiterating some points from the Testing and Success Measurement chapter in your Final Report.

A Report Structure for Software Projects

Conclusion - Generally speaking, section can take different forms - as a minimum you would normally provide a brief summary of your project work and a discussion of possible future work. You may also wish to reiterate the main outcomes of your project and give some idea of how you think the ideas dealt with by your project relate to real-world situations, etc.. For the Proposal, don't mention future work but do summarise your document.

References - Referencing is vital when building on the work of others. You may use any style of referencing but you should be consistent. All forms of plagiarism are taken extremely seriously. This is the first section that most markers will turn to when presented with a proposal or report.

A Report Structure for Software Projects

Appendices - Some information, for example program listings, is useful to include within the report for completeness, but would distract the reader from the flow of the discussion if it were included within the body of the document. Short extracts from major programs may be included to illustrate points but full program listings should only ever be placed within an appendix. Remember that the point of appendices is to make your report more readable. I don't expect to see apprentices in any Project Proposals.

Subjectively, I would hope not to see any in most reports but sometimes they're necessary.

Advice on Academic Writing

Avoid complexity, long sentences and embellishments

Focus on writing with precision and saying as much as is necessary to make your point

Structure and drafting are your friends