Experimental Computer Science and Computing Infrastructures

Miron Livny
Center for High Throughput Computing
Morgridge Institute for Research and
University of Wisconsin-Madison

MORGRIDGE
INSTITUTE FOR RESEARCH
AT THE UNIVERSITY OF WISCONSIN-MADISON

HT
CENTER FOR
HIGH THROUGHPUT
COMPUTING

THE UNIVERSITY of WISCONSIN
MADISON

P. Brady, the LIGO Data Analysis Software Chair, summarized the role of Condor technologies – "**Condor** manages LIGO compute-intensive data analysis jobs on more than 23,000 CPU slots offered by nine Linux clusters operated by the LSC. More than 250 LSC scientists **rely heavily** on Condor technologies to manage complex data analysis workflows. Over the years, LIGO and the Condor team have developed a strategic partnership resulting in many new software features that benefit LIGO and the entire Condor user community. We eagerly look forward to continuing this partnership since **Condor** technologies are **critical to the continued success** of the LIGO Data Grid as a platform for gravitational wave science."

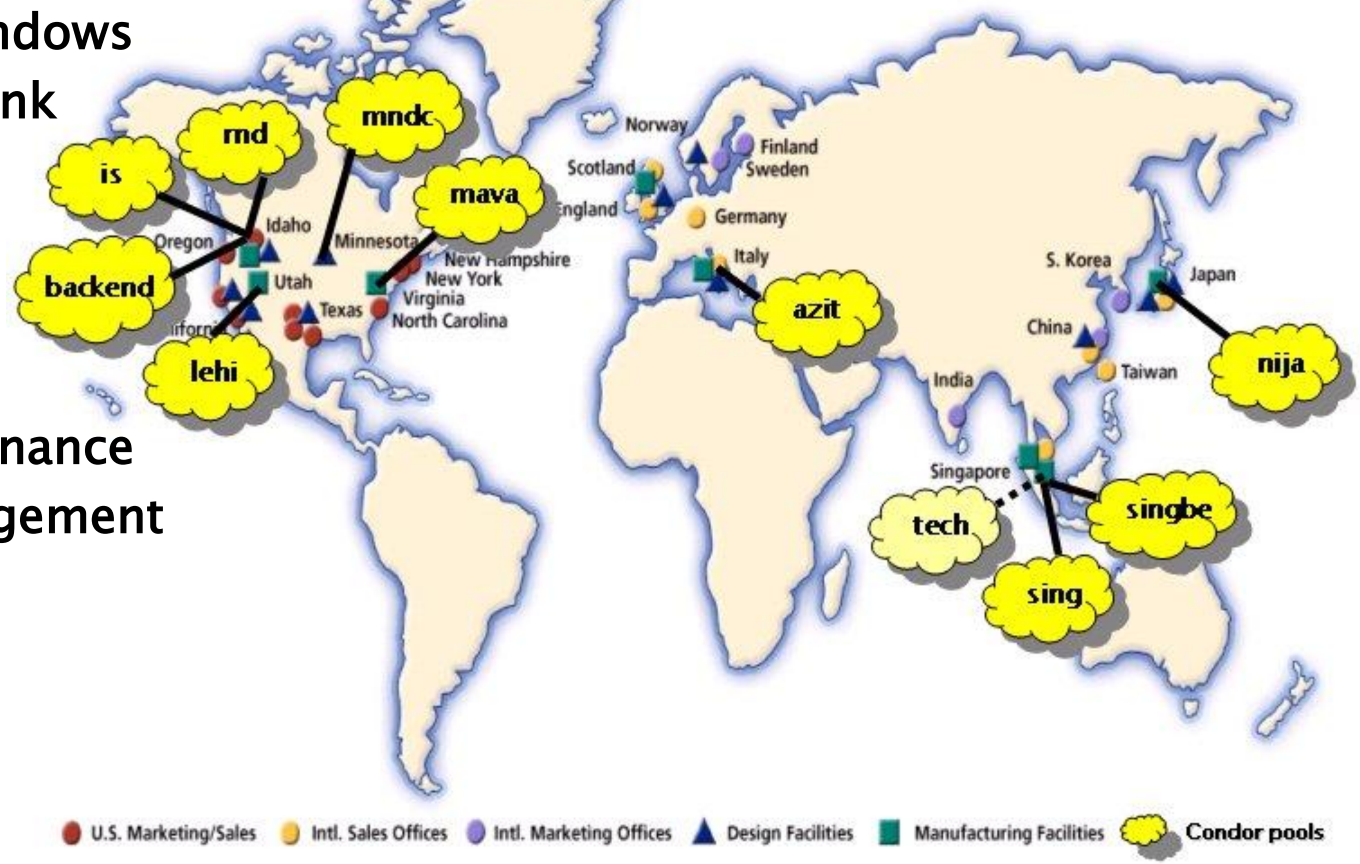
Condor at Micron

10,000+ processors in 12 “pools”
Linux, Solaris, Windows
<50th Top 500 Rank
3+ TeraFLOPS

Centralized governance
Distributed management

16+ applications
Self developed

Micron’s Global Grid



Slides used by UWCS with permission of Micron Technology, Inc.


redhat
CONGRATULATIONS
DR. MIRON LIVNY AND CHTC TEAM
FIRST RECIPIENT CLOUD LEADERSHIP AWARD


redhat.
2011
Red Hat Cloud Leadership Award
Center for High Throughput Computing
University of Wisconsin Madison

Copyright IBM Corp. 2000.© All rights reserved.

Trademarks and service marks

The following terms are trademarks of the IBM Corporation in the United States and/or other countries or both:

- AIX
- IBM
- LoadLeveler
- RISC System/6000
- RISC System/6000 Scalable POWERparallel Systems

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

LoadLeveler incorporates Condor, which was developed at the University of Wisconsin-Madison, and uses it with the permission of its authors.

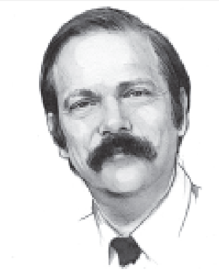

Academia
Enterprise
IT Comp.
Adoption

"Why are you leaving academia and taking a job in industry?"

"I want to have **impact!**"

In the words of Mike Carey

"I left academia for industry because I was drawn to the idea of getting more direct access to **real problems** - from customers and challenges encountered while building commercial-grade software - because I felt like I was in somewhat of a **mode of inventing and solving problems**, at least w.r.t. some of the things I'd been working on. Sure, that was leading to many written/submitted/accepted papers, but it was somehow less than satisfying after awhile."



DOI:10.1145/1810891.1810892

Moshe Y. Vardi

Science Has Only Two Legs

Science has been growing new legs of late. The traditional "legs" (or "pillars") of the scientific method were *theory* and *experimentation*. That was then. In 2005, for example, the U.S.

Presidential Information Technology Advisory Committee issued a report, "Computational Science: Ensuring America's Competitiveness," stating: "Together with theory and experimentation, computational science now

A scientific theory is an explanatory framework for a body of natural phenomena. A theory can be thought of as a model of reality at a certain level of abstraction. For a theory to be useful, it should explain existing observations

els. In system biology, for example, one often encounters computational models such as Petri Nets and Statecharts, which were developed originally in the context of computer science.

Computation has also always been

september 2010 | vol. 53 | no. 9 | **communications of the acm**

MORGRIDGE INSTITUTE FOR RESEARCH AT THE UNIVERSITY OF WISCONSIN-MADISON
HT CENTER FOR HIGH THROUGHPUT COMPUTING
THE UNIVERSITY of WISCONSIN MADISON

Edsger Dijkstra once stated:

"Computer science is no more about computers than astronomy is about telescopes."

Research Methods for Science By Michael P. Marder page 14. Published by Cambridge University Press

Abstract. We examine the philosophical disputes among computer scientists concerning methodological, ontological, and epistemological questions: Is **computer science** a branch of **mathematics, an engineering discipline, or a natural science**? Should knowledge about the behaviour of programs proceed deductively or empirically? Are computer programs on a par with mathematical objects, with mere data, or with mental processes? We conclude that distinct positions taken in regard to these questions emanate from distinct sets of received beliefs or paradigms within the discipline:

Eden, A. H. (2007). "Three Paradigms of Computer Science". *Minds and Machines* **17** (2): 135–167.

— **The rationalist paradigm**, which was common among theoretical computer scientists, defines computer science as a branch of mathematics, treats programs on a par with mathematical objects, and seeks certain, a priori knowledge about their 'correctness' by means of deductive reasoning.

— **The technocratic paradigm**, promulgated mainly by software engineers, defines computer science as an engineering discipline, treats programs as mere data, and seeks probable, a posteriori knowledge about their reliability empirically using testing suites.

— **The scientific paradigm**, prevalent in the branches of artificial intelligence, defines computer science as a **natural (empirical) science**, takes programs to be entities on a par with mental processes, and seeks a priori and a posteriori knowledge about them by combining formal deduction and **scientific experimentation**.

Eden, A. H. (2007). "Three Paradigms of Computer Science". *Minds and Machines* **17** (2): 135–167.

Who are we?

Part of a Computer Sciences department (ranked 11th in the US), have been working on distributed computing since the early 80's and have been collaborating with domain scientists since the late 80's. So far we failed to engage any (OK, maybe very few!) other faculty from the department (or other universities) in our infrastructure and software engineering problems/challenges. So all we know and do is self-taught and the result of ongoing experimental work.

" … Since the early days of mankind the primary motivation for the establishment of *communities* has been the idea that by being part of an organized group the capabilities of an individual are improved. The great progress in the area of inter-computer communication led to the development of means by which stand-alone processing sub-systems can be integrated into multi-computer '*communities*'. … "

Miron Livny, " *Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems*.", Ph.D thesis, July 1983.

The words of Koheleth son of David, king in Jerusalem ~ 200 A.D.

Only that shall happen
Which has happened,
Only that occur
Which has occurred;
There is nothing new
Beneath the sun!

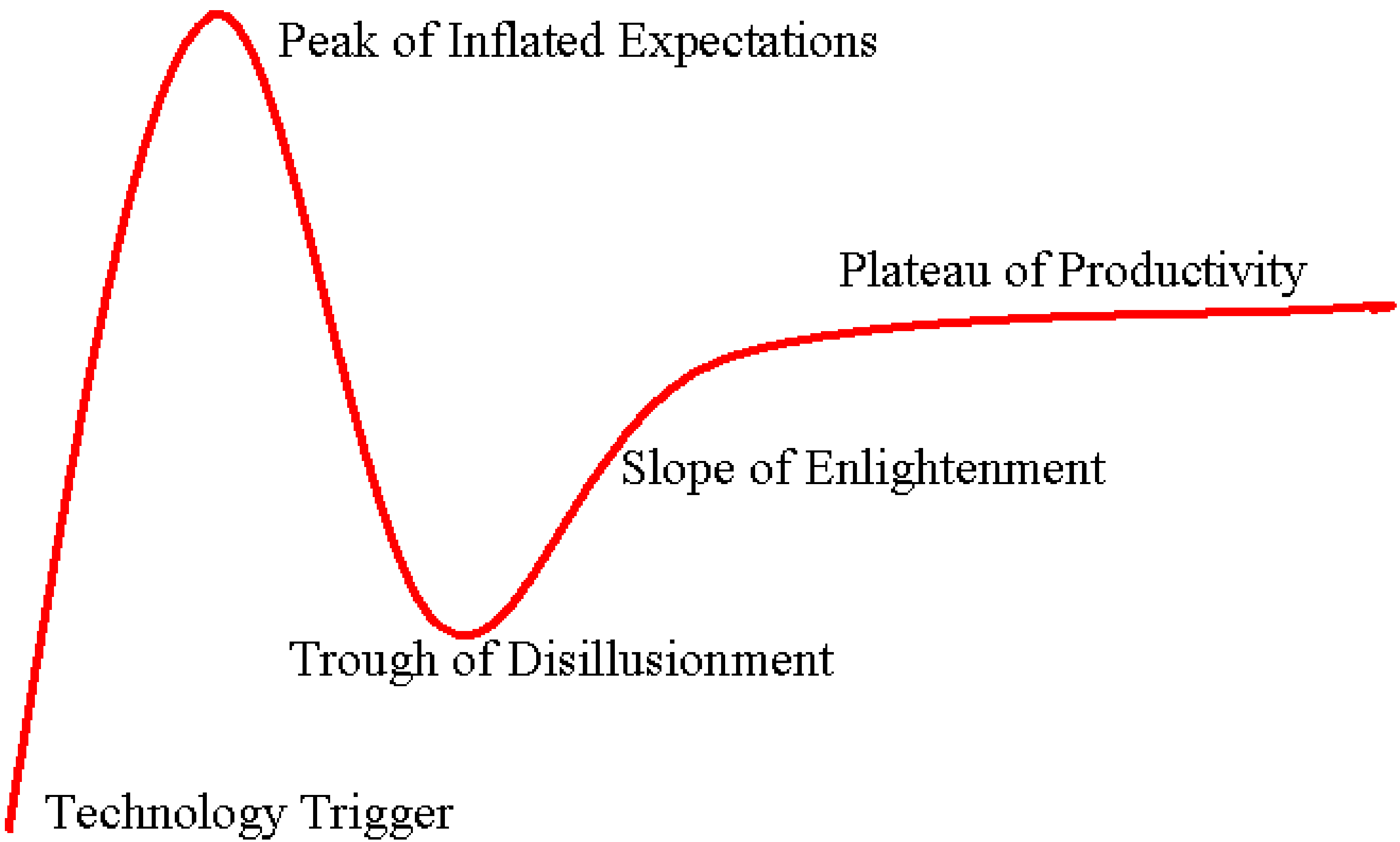
Ecclesiastes, (*Kohelet*, "son of David, and king in Jerusalem" alias Solomon, Wood engraving
Gustave Doré (1832–1883)

Ecclesiastes Chapter 1 verse 9

MORGRIDGE INSTITUTE FOR RESEARCH AT THE UNIVERSITY OF WISCONSIN-MADISON

THE UNIVERSITY of WISCONSIN MADISON

Gartner Hype Cycle



Perspectives on Grid Computing

Uwe Schwiegelshohn Rosa M. Badia Marian Bubak Marco Danelutto
Schahram Dustdar Fabrizio Gagliardi Alfred Geiger Ladislav Hluchy
Dieter Kranzlmüller Erwin Laure Thierry Priol Alexander Reinefeld
Michael Resch Andreas Reuter Otto Rienhoff Thomas Rüter Peter Sloot
Domenico Talia Klaus Ullmann Ramin Yahyapour Gabriele von Voigt

We should not waste our time in redefining terms or key technologies: clusters, Grids, Clouds... What is in a name? Ian Foster recently quoted Miron Livny saying: **"I was doing Cloud computing way before people called it Grid computing"**, referring to the ground breaking Condor technology. It is the Grid scientific paradigm that counts!

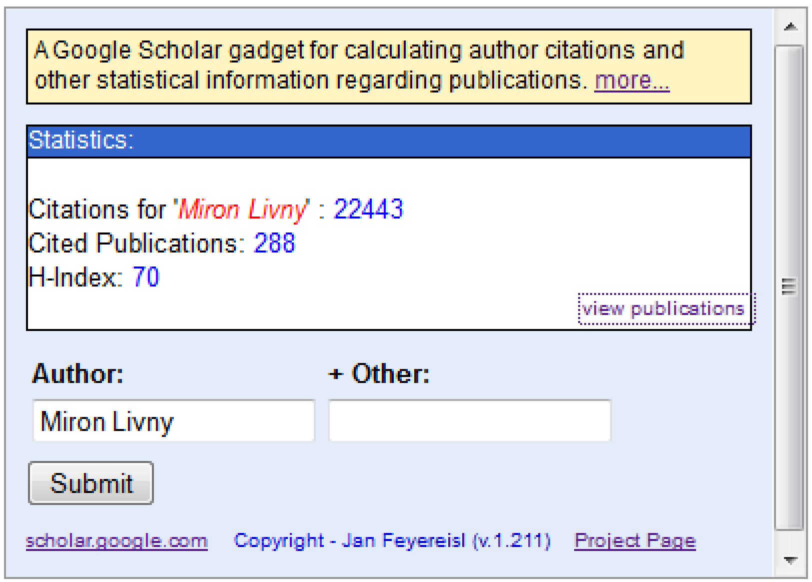
(my) terminology

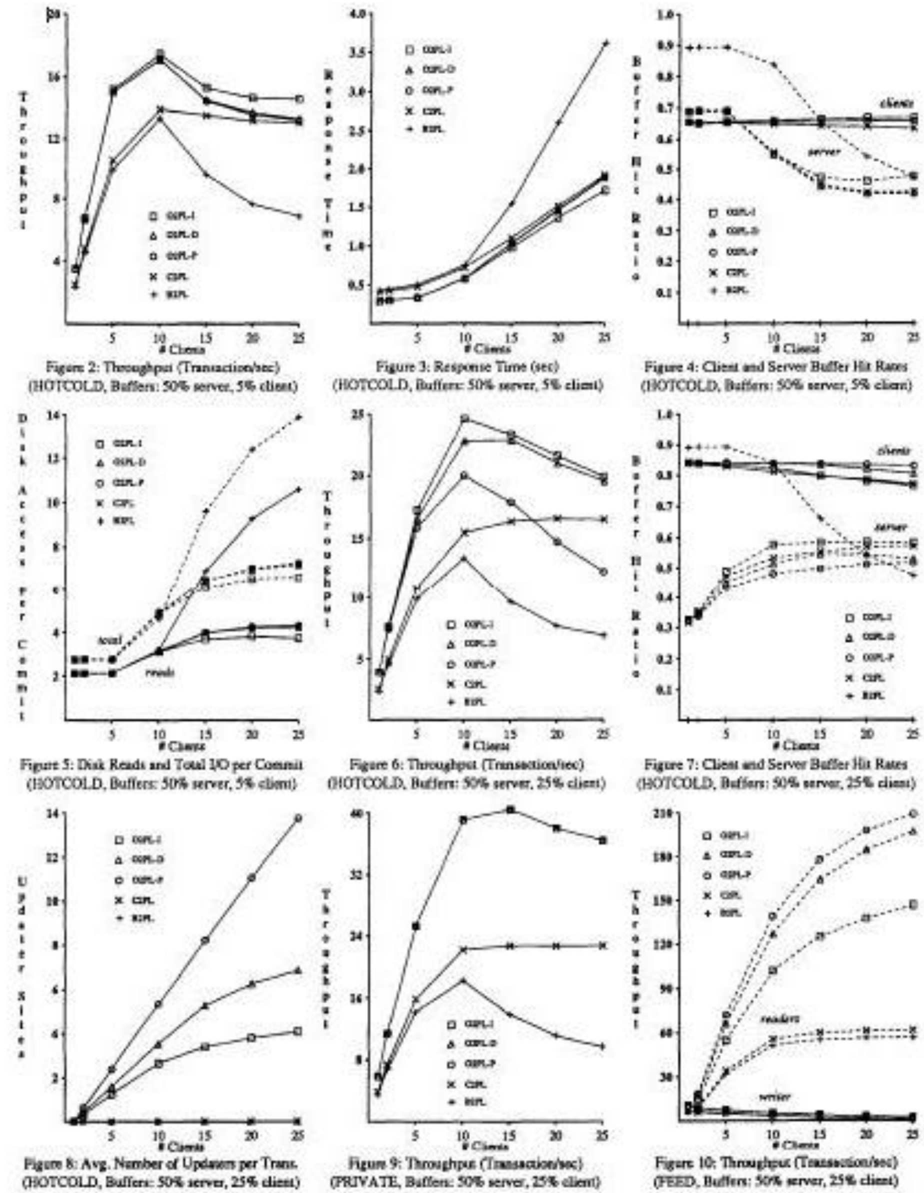
- **Experiment** - an act or operation for the purpose of discovering something unknown or of testing a principle, supposition, etc.:
- **Technology Adoption** – to select a technology as a means to meet an ends of significant importance/value
- **Real users** – individuals or groups who adopt (and use) a computing technology
- **Experimental Computer Science** – advance the state of the art of computing (new frameworks, new technologies, new abstractions) through experiments that involve real users

Condor Team 2010

Established 1985

We know how to play the paper ‘game’ …





363

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

The MapReduce implementation relies on an in-house cluster management system that is responsible for distributing and running user tasks on a large collection of shared machines. Though not the focus of this paper, the cluster management system is similar in spirit to other systems such as Condor [16].

BAD-FS [5] has a very different programming model from MapReduce, and unlike MapReduce, is targeted to the execution of jobs across a wide-area network. However, there are two fundamental similarities. (1) Both systems use redundant execution to recover from data loss caused by failures. (2) Both use locality-aware scheduling to reduce the amount of data sent across congested network links.

Abstract

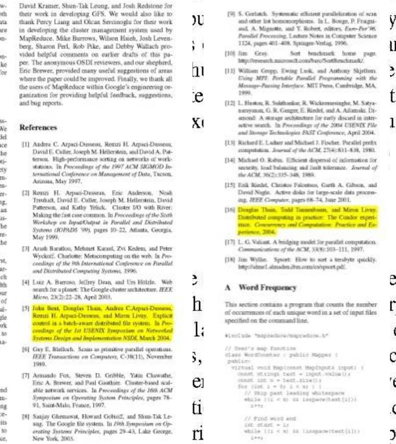
MapReduce is a programming model
ated implementation for processing and
data sets. Users specify a *map* function
key/value pair to generate a set of interm
pairs, and a *reduce* function that merges
values associated with the same interme
real world tasks are expressible in this n
in the paper.

Programs written in this functional sty
cally parallelized and executed on a large
modity machines. The run-time system t
details of partitioning the input data, sch
gram's execution across a set of machine
chine failures, and managing the require
communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Ou ... M ... large
clust ... able:
a typ ... y ter-
abyte ... mers
find t ... pro-
gram ... thou-
sand ... sters
every

1

Over ... ers at
Goog ... rpose
comp ... data,
such ... c., to
comp ... erted
indic ... cture
of w ... pages
crawled per host, the set of ... queries in a



David Kramer, Shun-Tak Leung, and Josh Redstone for their work in developing GFS. We would also like to thank Percy Liang and Olcan Sercinoglu for their work in developing the cluster management system used by MapReduce. Mike Burrows, Wilson Hsieh, Josh Levenberg, Sharon Perl, Rob Pike, and Debby Wallach provided helpful comments on earlier drafts of this paper. The anonymous OSDI reviewers, and our shepherd, Eric Brewer, provided many useful suggestions of areas where the paper could be improved. Finally, we thank all the users of MapReduce within Google's engineering organization for providing helpful feedback, suggestions, and bug reports.

References

[1] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. High-performance sorting on networks of workstations. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, May 1997.

[2] Remzi H. Arpaci-Dusseau, Eric Anderson, Noah Treuhaft, David E. Culler, Joseph M. Hellerstein, David Patterson, and Kathy Yelick. Cluster I/O with River: Making the fast case common. In *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems (IOPADS '99)*, pages 10–22, Atlanta, Georgia, May 1999.

[3] Arash Baratloo, Mehmet Karaul, Zvi Kedem, and Peter Wyckoff. Charlotte: Metacomputing on the web. In *Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems*, 1996.

[4] Luiz A. Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23(2):22–28, April 2003.

[5] John Bent, Douglas Thain, Andrea C.Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Miron Livny. Explicit control in a batch-aware distributed file system. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation NSDI*, March 2004.

[6] Guy E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, C-38(11), November 1989.

[7] Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer, and Paul Gauthier. Cluster-based scalable network services. In *Proceedings of the 16th ACM Symposium on Operating System Principles*, pages 78–91, Saint-Malo, France, 1997.

[8] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In *19th Symposium on Operating Systems Principles*, pages 29–43, Lake George, New York, 2003.

[9] S. Gorlatch. Systematic efficient parallelization of scan and other list homomorphisms. In L. Bouge, P. Fraigniaud, A. Mignotte, and Y. Robert, editors, *Euro-Par'96. Parallel Processing*, Lecture Notes in Computer Science 1124, pages 401–408. Springer-Verlag, 1996.

[10] Jim Gray. Sort benchmark home page. http://research.microsoft.com/barc/SortBenchmark/.

[11] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, Cambridge, MA, 1999.

[12] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *Proceedings of the 2004 USENIX File and Storage Technologies FAST Conference*, April 2004.

[13] Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, 1980.

[14] Michael O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[15] Erik Riedel, Christos Faloutsos, Garth A. Gibson, and David Nagle. Active disks for large-scale data processing. *IEEE Computer*, pages 68–74, June 2001.

[16] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The Condor experience. *Concurrency and Computation: Practice and Experience*, 2004.

[17] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1997.

[18] Jim Wyllie. Spsort: How to sort a terabyte quickly. http://alme1.almaden.ibm.com/cs/spsort.pdf.

A Word Frequency

This section contains a program that counts the number of occurrences of each unique word in a set of input files specified on the command line.

```
#include "mapreduce/mapreduce.h"

// User's map function
class WordCounter : public Mapper {
 public:
  virtual void Map(const MapInput& input) {
    const string& text = input.value();
    const int n = text.size();
    for (int i = 0; i < n; ) {
      // Skip past leading whitespace
      while ((i < n) && isspace(text[i]))
        i++;

      // Find word end
      int start = i;
      while ((i < n) && !isspace(text[i]))
        i++;
```

and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then
apply
the sa
propr
specif
lelize
as the

The
powe
and d
with
high p

Sec
gives
menta
our cl
scribe
that w
measurements of our implementation for a variety of tasks. Section 6 explores the use of MapReduce within Google including our experiences in using it as the basis

High Throughput Computing

We first introduced the distinction between High Performance Computing (HPC) and High Throughput Computing (HTC) in a seminar at the NASA Goddard Flight Center in July of **1996** and a month later at the European Laboratory for Particle Physics (CERN). In June of 1997 HPCWire published an interview on High Throughput Computing.

```
HIGH THROUGHPUT COMPUTING: AN INTERVIEW WITH MIRON LIVNY               06.27.97
by Alan Beck, editor in chief                                            HPCwire
==============================================================================

  This month, NCSA's (National Center for Supercomputing Applications)
Advanced Computing Group (ACG) will begin testing Condor, a software system
developed at the University of Wisconsin that promises to expand computing
capabilities through efficient capture of cycles on idle machines. The
software, operating within an HTC (High Throughput Computing) rather than a
traditional HPC (High Performance Computing) paradigm, organizes machines
```

Why HTC?

For many experimental scientists, scientific progress and quality of research are strongly linked to computing throughput. In other words, they are less concerned about instantaneous computing power. Instead, what matters to them is the amount of computing they can harness over a month or a year --- they measure computing power in units of scenarios per day, wind patterns per week, instructions sets per month, or crystal configurations per year.

High Throughput Computing
is a
24-7-365
activity

FLOPY ≠ *(60*60*24*7*52)*FLOPS*

Obstacles to HTC

- Ownership Distribution (Sociology)
- Customer Awareness (Education)
- Size and Uncertainties (Robustness)
- Technology Evolution (Portability)
- Physical Distribution (Technology)

" … We claim that these **mechanisms**, although originally developed in the context of a cluster of workstations, are also applicable to computational **grids**. In addition to the required flexibility of services in these grids, a very important concern is that the system be **robust** enough to run in "**production mode**" continuously even in the face of component failures. … "

Miron Livny & Rajesh Raman, *"High Throughput Resource Management"*, **in "*The Grid: Blueprint for a New Computing Infrastructure*".**

MORGRIDGE INSTITUTE FOR RESEARCH AT THE UNIVERSITY OF WISCONSIN-MADISON

HT CENTER FOR HIGH THROUGHPUT COMPUTING

THE UNIVERSITY of WISCONSIN MADISON

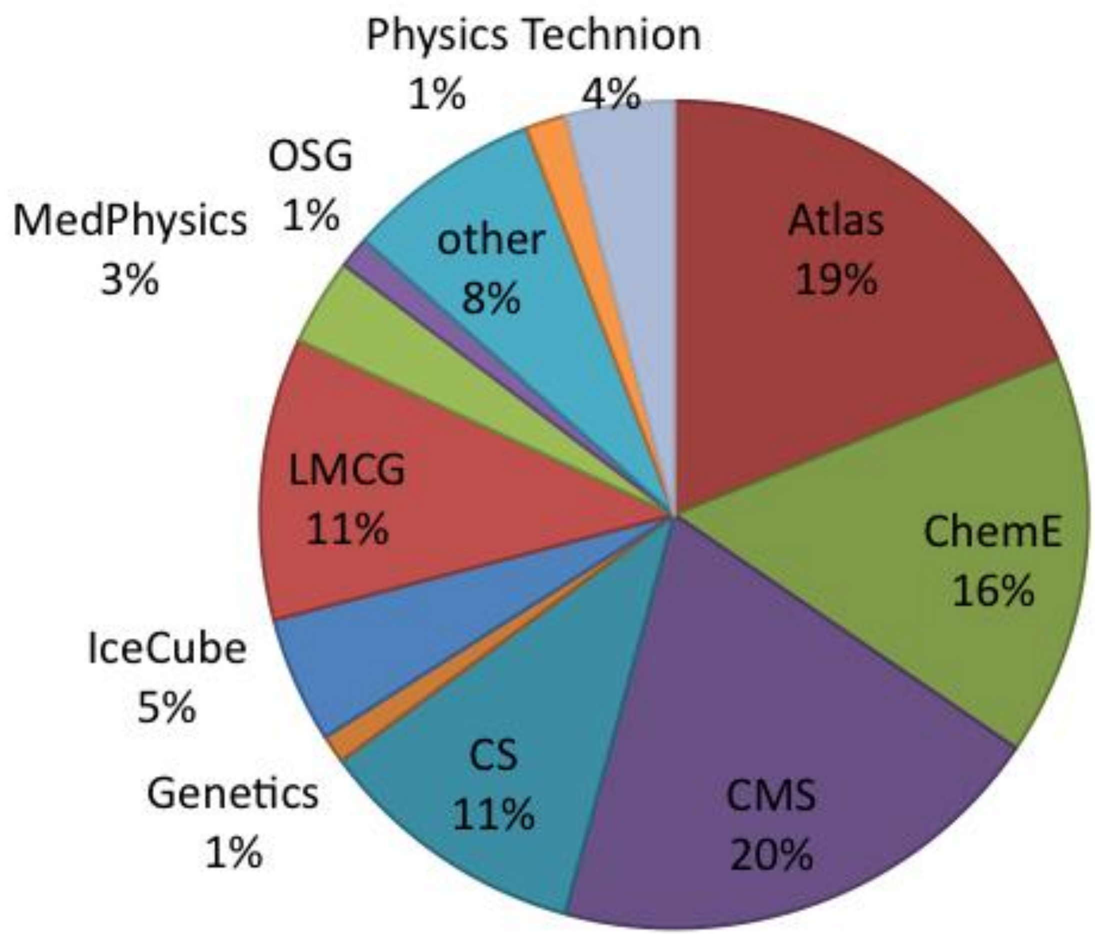
Main Threads of Activities

- **Distributed Computing Research** – develop and evaluate new concepts, frameworks and technologies
- Keep the Condor system "**flight worthy**" and support our users
- **The Grid Laboratory Of Wisconsin (GLOW)** – build, maintain and operate a distributed computing and storage infrastructure on the UW campus
- **The Open Science Grid (OSG)** – build and operate a national distributed computing and storage infrastructure
- **The NSF Middleware Initiative (NMI)** – develop, build and operate a national Build and Test facility

"The members of the OSG are united by a commitment to promote the adoption and to advance the state of the art of *distributed high throughput computing (DHTC) – shared utilization of autonomous* resources where all the elements are optimized for maximizing computational throughput."

Open Science Grid (OSG)
DHTC at the National Level

Grid Laboratory of Wisconsin (GLOW) HTC at the campus level

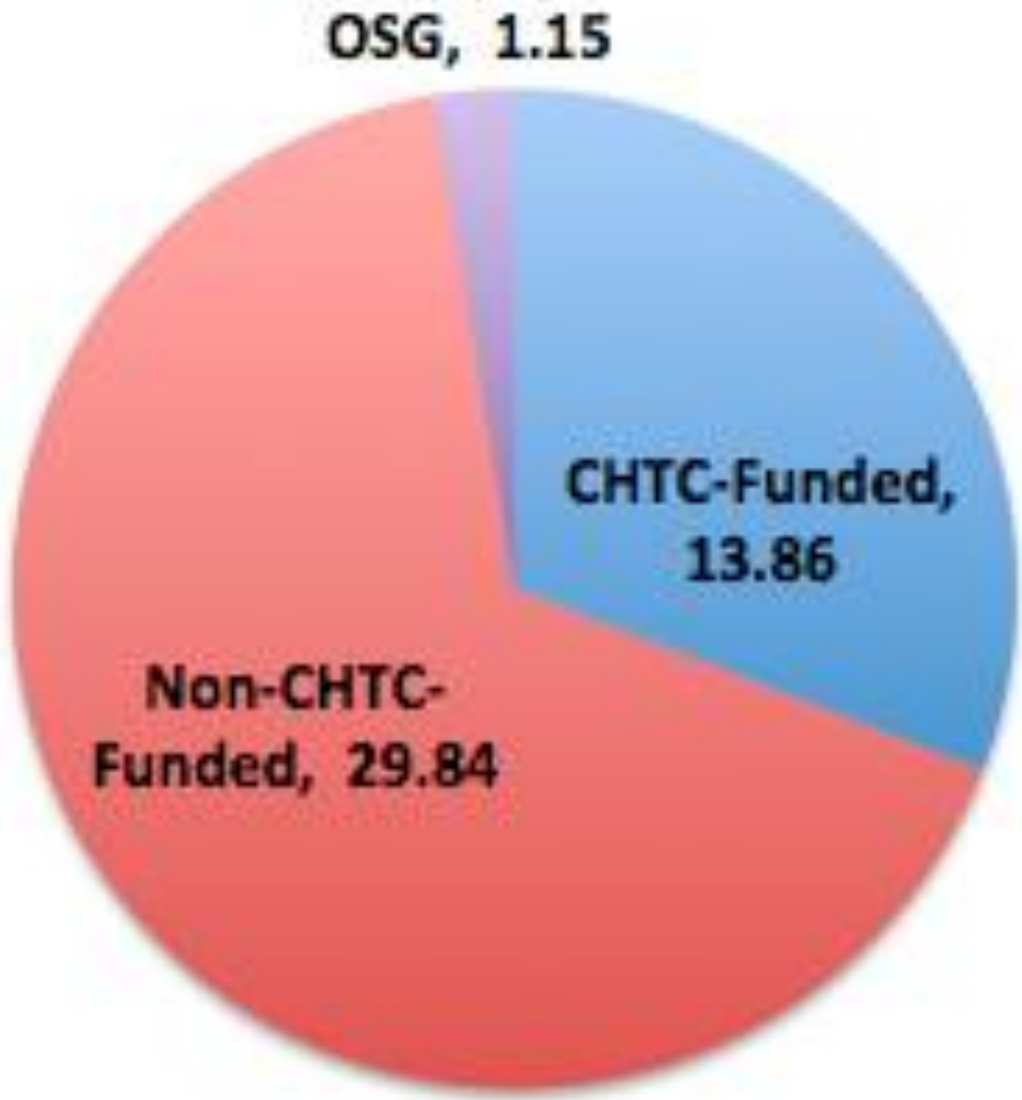


Usage 04/04-04/10

114M Hours

GLOW (CHTC) Cycles delivered over the past year

45M total



Some Condor software Numbers

Over the past year every month we have:

- Released a new version of Condor to the public
- Performed over 170 commits to the codebase
- Modified over 350 source code files
- Changed over 8.5K lines of code (Condor source code written at UW-Madison as of June 2011 sits at 922K LOC)
- Compiled about 2.5K builds of the code for testing purposes
- Ran 930K regression tests (functional and unit)

We need more and higher quality Experimental Computer Science!

Case 1: 10,000 Cores "Tanuki"

- Run time = 8 hours
- 1.14 compute-years of computing executed every hour
- Cluster Time = 80,000 hours = 9.1 compute years.
- Total run time cost = ~$8,500

- 1250 c1.xlarge ec2 instances (8 cores / 7-GB RAM)
- 10,000 cores, 8.75 TB RAM, 2 PB of disk space
- Weighs in at number 75 of Top 500 SuperComputing list
- Cost to run = ~ $1,060 / hour

Customer Goals

- Genentech: “Examine how proteins bind to each other in research that may lead to medical treatments.”

  - www.networkworld.com

- Customer wants to test the scalability of CycleCloud: “Can we run 10,000 jobs at once?”
- Same workflow would take weeks or months on existing internal infrastructure.

Run Timeline

- 12:35 – 10,000 Jobs submitted and requests for batches cores are initiated
- 12:45 – 2,000 cores acquired
- 1:18 – 10,000 cores acquired
- 9:15 – Cluster shut down

$1,279-per-hour, 30,000-core cluster built on Amazon EC2 cloud

By Jon Brodkin | Published 22 days ago

A vendor called Cycle Computing is on a mission to demonstrate the potential of Amazon's cloud by building increasingly large clusters on the Elastic Compute Cloud. Even with Amazon, building a cluster takes some work, but Cycle combines several technologies to ease the process and recently used them to create a 30,000-core cluster running CentOS Linux.

The cluster, announced publicly this week, was created for an unnamed "Top 5 Pharma" customer, and ran for about seven hours at the end of July at a peak cost of $1,279 per hour, including the fees to Amazon and Cycle Computing. The details are impressive: 3,809 compute instances, each with eight cores and 7GB of RAM, for a total of 30,472 cores, 26.7TB of RAM and 2PB (petabytes) of disk space. Security was ensured with HTTPS, SSH and 256-bit AES encryption, and the cluster ran across data centers in three Amazon regions in the United States and Europe. The cluster was dubbed "Nekomata."