From Principles to Capabilities - the Birth and Evolution of High Throughput Computing

Miron Livny

Wisconsin Institutes for Discovery

Madison-Wisconsin

MORGRIDGE INSTITUTE FOR RESEARCH

CENTER FOR HIGH THROUGHPUT COMPUTING

WISCONSIN INSTITUTE FOR DISCOVERY

The lessons of the past
and
the illusions of predictions

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

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

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

Ecclesiastes Chapter 1 verse 9

The Talmud says in the name of Rabbi Yochanan,

"Since the destruction of the Temple, prophecy has been taken from prophets and given to fools and children."

(Baba Batra 12b)

In 1996 I introduced the distinction between High **Performance** Computing (HPC) and High **Throughput** Computing (**HTC**) in a seminar at the NASA Goddard Flight Center in 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 day, 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*

MORGRIDGE INSTITUTE FOR RESEARCH

CENTER FOR HIGH THROUGHPUT COMPUTING

WISCONSIN INSTITUTE FOR DISCOVERY

"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."

Scientific Computing for the 21st Century

Workshop on HPC and Super-computing for
Future Science Applications

June 6, 2013

Richard Carlson
Richard.Carlson@science.doe.gov

Traditional Scientific Computing Issues

- **Tussle between High Performance Computing and High Throughput Computing**
  - Capability vs Capacity
- **Tussle between Grid / Cloud / Distributed computing**
  - What are the differences between grid and cloud
- **Tussle between hardware ownership and software services**
  - Who owns and manages the hardware vs the deployed services
- **Tussle between basic research and sustained deployment activities**
  - How to balance research with sustainability

In 1978 I fell in love with the problem of load balancing in distributed systems

Claims for "benefits" provided by Distributed Processing Systems

P.H. Enslow, *"What is a Distributed Data Processing System?"* Computer, January 1978

- High Availability and Reliability
- High System Performance
- Ease of Modular and Incremental Growth
- Automatic Load and Resource Sharing
- Good Response to Temporary Overloads
- Easy Expansion in Capacity and/or Function

Definitional Criteria for a Distributed Processing System

P.H. Enslow and T. G. Saponas "*"Distributed and Decentralized Control in Fully Distributed Processing Systems"* Technical Report, 1981

- Multiplicity of resources
- Component interconnection
- **Unity of control**
- System transparency
- **Component autonomy**

Unity of Control

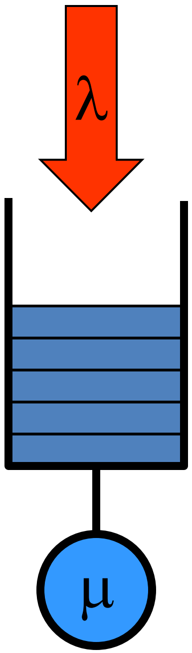
All the component of the system should be **unified** in their desire to achieve a **common goal**. This goal will determine the rules according to which each of these elements will be controlled.

Component autonomy

The components of the system, both the logical and physical, should be **autonomous** and are thus afforded the ability to refuse a request of service made by another element. However, in order to achieve the system’s goals they have to interact in a **cooperative** manner and thus adhere to a common set of policies. These policies should be carried out by the control schemes of each element.

It is always a tradeoff

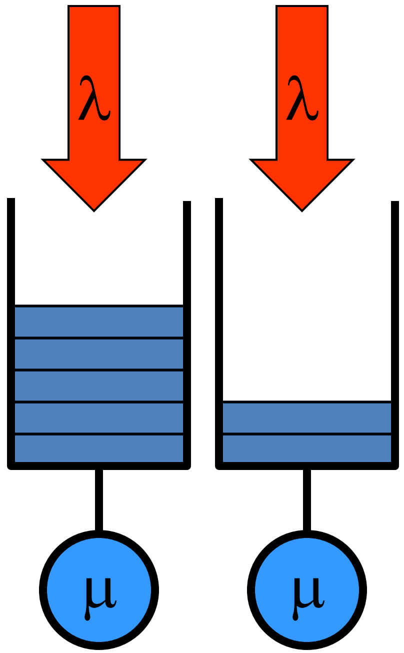
BASICS OF A M/M/1 SYSTEM



Expected # of customers is $1/(1-\rho)$, where ($\rho = \lambda/\mu$) is the utilization

When utilization is 80%, you wait on the average 4 units for every unit of service

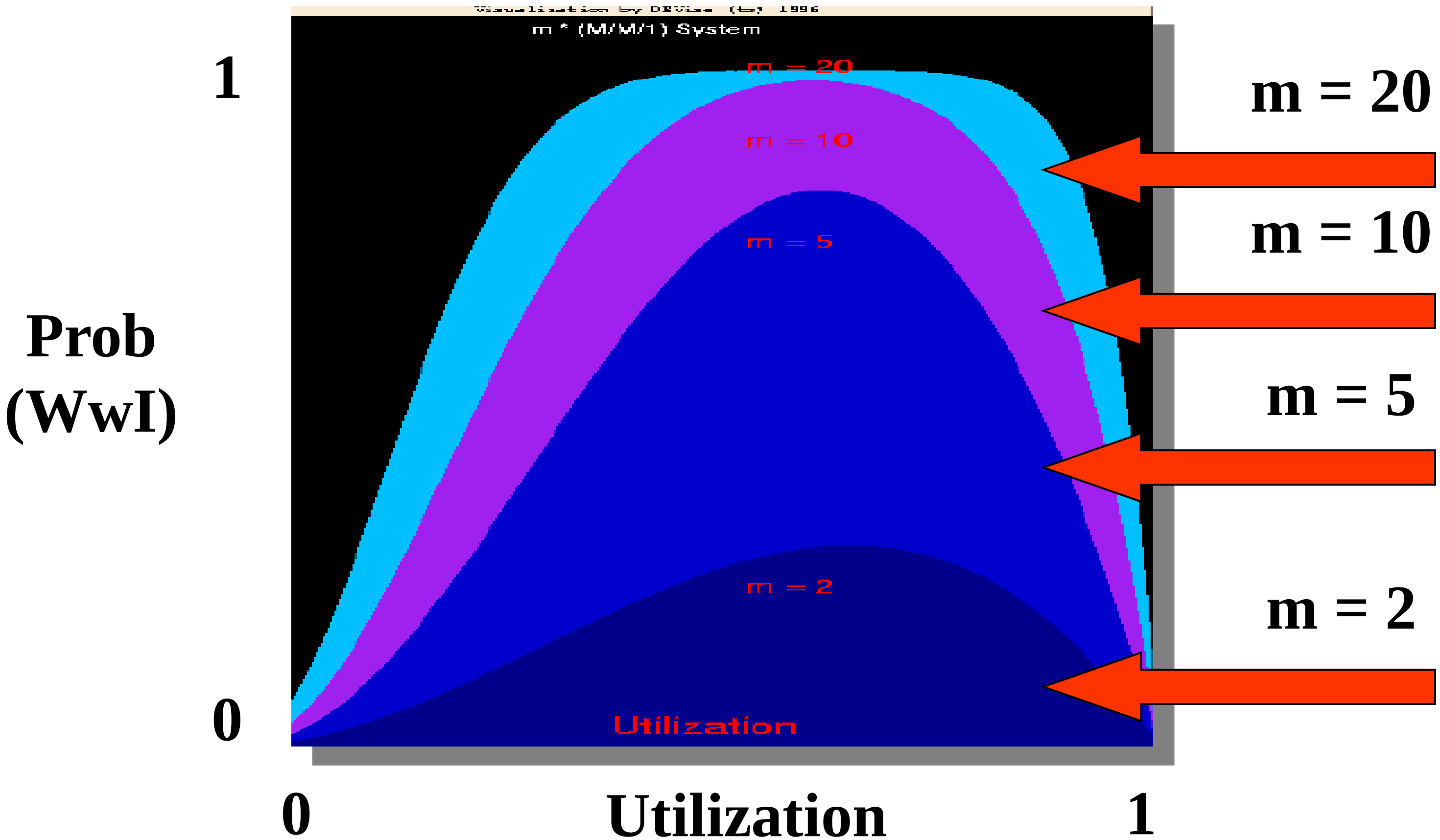
BASICS OF TWO M/M/1 SYSTEMS



When utilization is 80%, you wait on the average 4 units for every unit of service

When utilization is 80%, 25% of the time a customer is waiting for service while a server is idle

Wait while Idle (WwI) in m*M/M/1



In 1983 I wrote a Ph.D. thesis –

"Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems"

http://www.cs.wisc.edu/condor/doc/livny-dissertation.pdf

Should I stay or should I move?

“ … 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.

In 1985 we extended the scope of the distributed load balancing problem to include "ownership" of resources

Should I share and if I do with whom and when?

Now you have customers who are consumers, providers or both

What Did We Learn From Serving a Quarter of a Million Batch Jobs on a Cluster of Privately Owned Workstations

1992

Miron Livny

Computer Sciences Department
University of Wisconsin — Madison
Madison, Wisconsin
{miron@cs.wisc.edu}

User Prospective

- Maximize the capacity of resources accessible via a single interface
- Minimize overhead of accessing remote capacity
- Preserve local computation environment

Submit Locally and run Globally

(Here is the work and here are the resources I bring to the table)

Global Scientific Computing via a Flock of Condors

CERN 92

Miron Livny

Computer Sciences Department
University of Wisconsin — Madison
Madison, Wisconsin
{miron@cs.wisc.edu}

MISSION

Give scientists effective and efficient access to large amounts of cheap (if possible free) CPU cycles and main memory storage

THE CHALLENGE

How to turn existing privetly owned clusters of *workstations*, *farms*, *multiprocessors*, and *supercomputers* into an efficient and effective Global Computing Environment?

In other words, how to minimize wait while idle?

APPROACH

Use wide-area networks to transfer batch jobs between Condor systems

- Boundaries of each Condor system will be determined by physical or administrative considerations

TWO EFFORTS

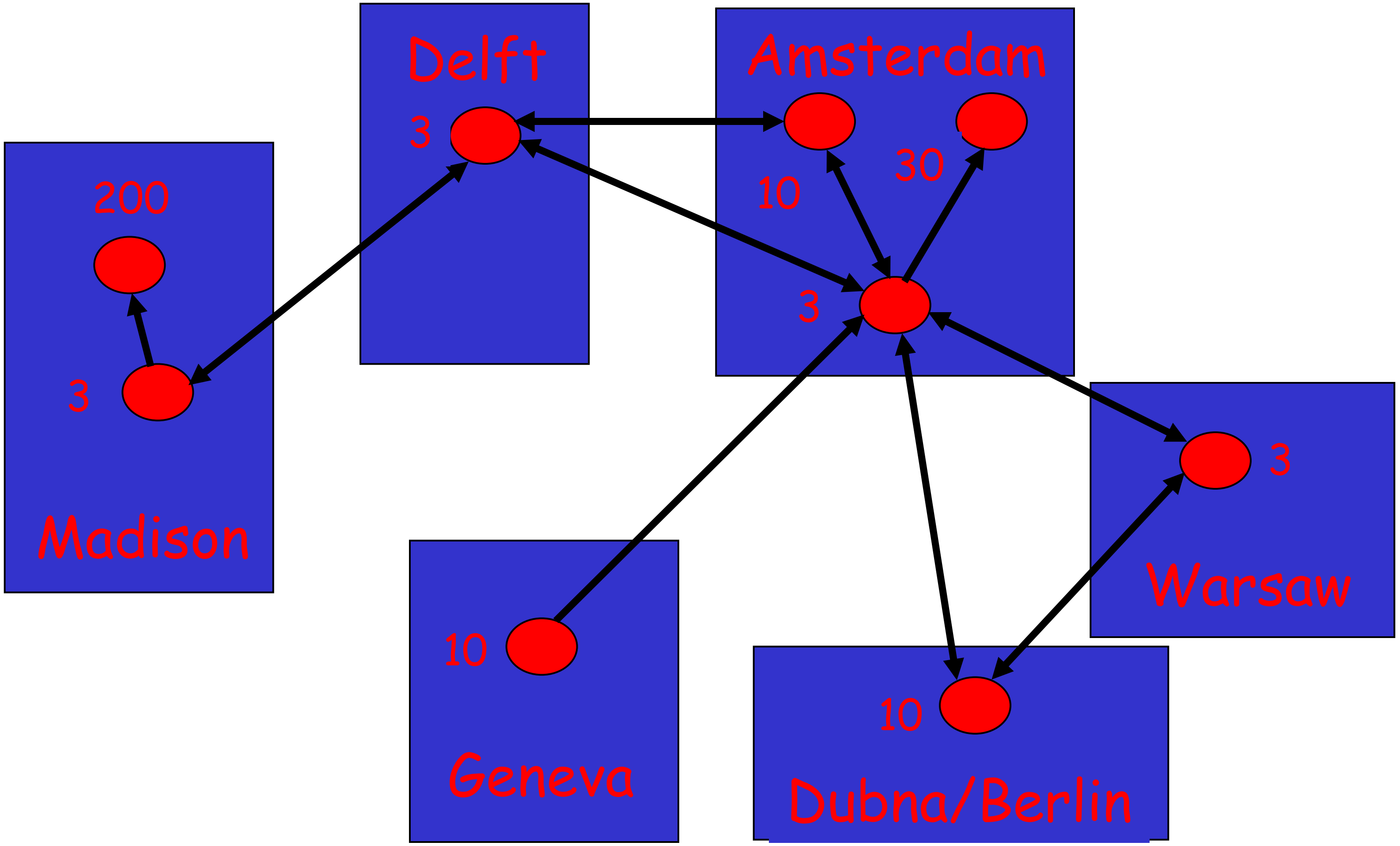
☐ ***UW CAMPUS***

Condor systems at Engineering, Statistics, and Computer Sciences

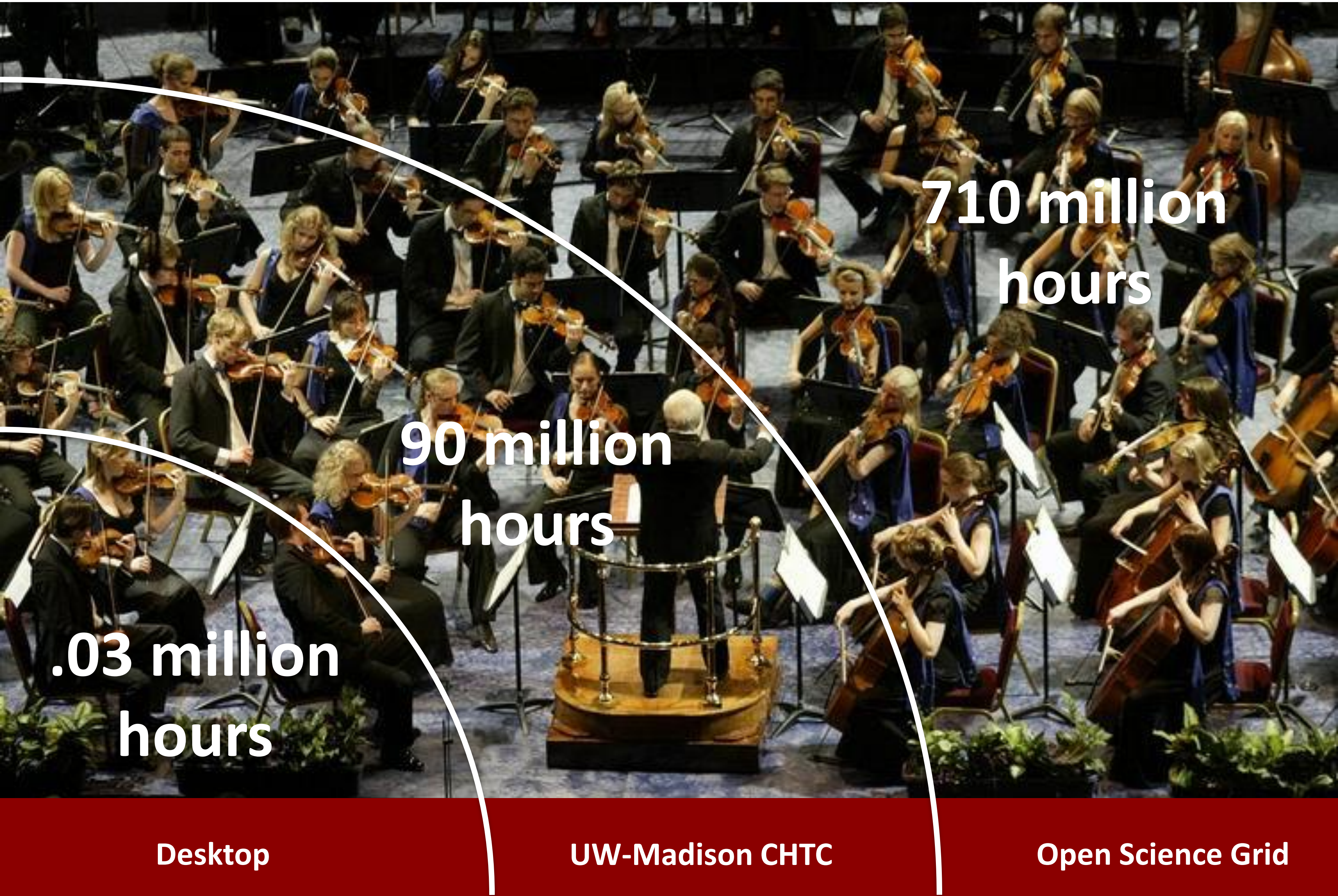
☐ ***INTERNATIONAL***

We have started a collaboration between CERN-SMC-NIKHEF-Univ. of Amsterdam, and University of Wisconsin-Madison

1994 Worldwide Flock of Condors



HTC on the UW campus



Subject: Meeting request

From: Michael Gofman <michael.gofman@gmail.com>

Date: Thu, 16 May 2013 11:47:50 -0500

To: MIRON LIVNY <MIRON@cs.wisc.edu>

Dear Miron,

I am an assistant professor of finance at UW-Madison. I did my Phd at the University of Chicago and master degrees at the Tel Aviv University.

In the last couple months I was using HTC resources that you developed to compute optimal financial architecture.

I would like to meet with you and tell you more about my project as well to thank you personally for developing this amazing platform.

Yours,

Michael

Experimental Computer Science where you and other scientists are the

MORGRIDGE INSTITUTE FOR RESEARCH

CENTER FOR HIGH THROUGHPUT COMPUTING

WISCONSIN INSTITUTE FOR DISCOVERY

A Google Scholar gadget for calculating author citations and other statistical information regarding publications. more...

Statistics:

Citations for '*Miron Livny*' : 22443
Cited Publications: 288
H-Index: 70

view publications

Author: Miron Livny

+ Other:

Submit

2006 SIGMOD Test of Time Award

BIRCH: An Efficient Data Clustering Method for Very Large Databases

Tian Zhang (University of Wisconsin, Madison), Raghu Ramakrishnan (University of Wisconsin, Madison), and Miron Livny (University of Wisconsin, Madison)

The paper introduces a novel, scalable, simple yet effective technique for clustering large multi-dimensional datasets, based on core database management system technology (indexing). It has had significant research impact and has influenced commercial products.

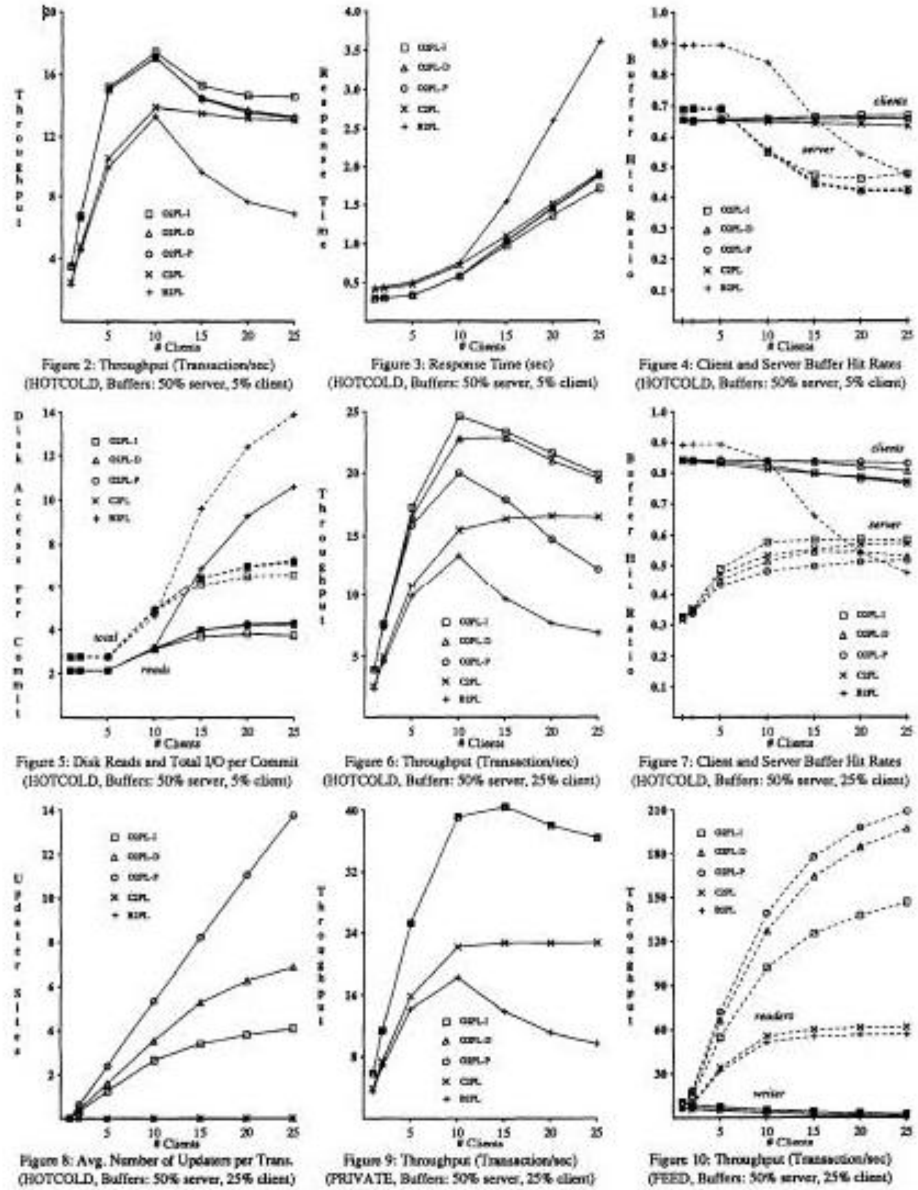


Figure 2: Throughput (Transaction/sec) (HOTCOLD, Buffers: 50% server, 5% client)
Figure 3: Response Time (sec) (HOTCOLD, Buffers: 50% server, 5% client)
Figure 4: Client and Server Buffer Hit Rates (HOTCOLD, Buffers: 50% server, 5% client)
Figure 5: Disk Reads and Total I/O per Commit (HOTCOLD, Buffers: 50% server, 5% client)
Figure 6: Throughput (Transaction/sec) (HOTCOLD, Buffers: 50% server, 25% client)
Figure 7: Client and Server Buffer Hit Rates (HOTCOLD, Buffers: 50% server, 25% client)
Figure 8: Avg. Number of Updaters per Trans. (HOTCOLD, Buffers: 50% server, 25% client)
Figure 9: Throughput (Transaction/sec) (PRIVATE, Buffers: 50% server, 25% client)
Figure 10: Throughput (Transaction/sec) (FEED, Buffers: 50% server, 25% client)

Dear Professor Livny,

I'm writing to you as I wish to invite you to a panel we're organizing at the next ECCS 2012 on "Experiments in Computer Science: Are Traditional Experimental Principles Enough?"

I was present during your ECSS presentation last year in Milan on "Experimental Computer Science and Computing Infrastructures" and, actually, was the person who asked you about a more scientifically oriented notion of experiment.

I must confess that your talk, and the discussion I had with some collegues after, was ones of the driving forces behind the organization of this panel and a pre-summit workshop (also on experiments in computer science So it would be really fantastic if you would be interested in participating in the panel.

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 behavior 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.

Real and hard Computer Science problems are exposed when you do it for “real”

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

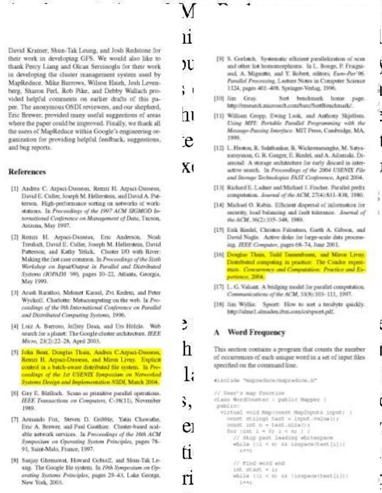
MapReduce is a programming model
ated implementation for processing and
data sets. Users specify a *map* function
key/value pair to generate a set of interm
pairs, and a *reduce* function that merges
values associated with the same interme
real world tasks are expressible in this m
in the paper.

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.

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].

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

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.


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++;
```

tasks. Section 6 explores the use of MapReduce within Google including our experiences in using it as the basis

You have Impact!

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

“I want to have

impact!”

MORGRIDGE INSTITUTE FOR RESEARCH

CENTER FOR HIGH THROUGHPUT COMPUTING

WISCONSIN INSTITUTE FOR DISCOVERY

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."

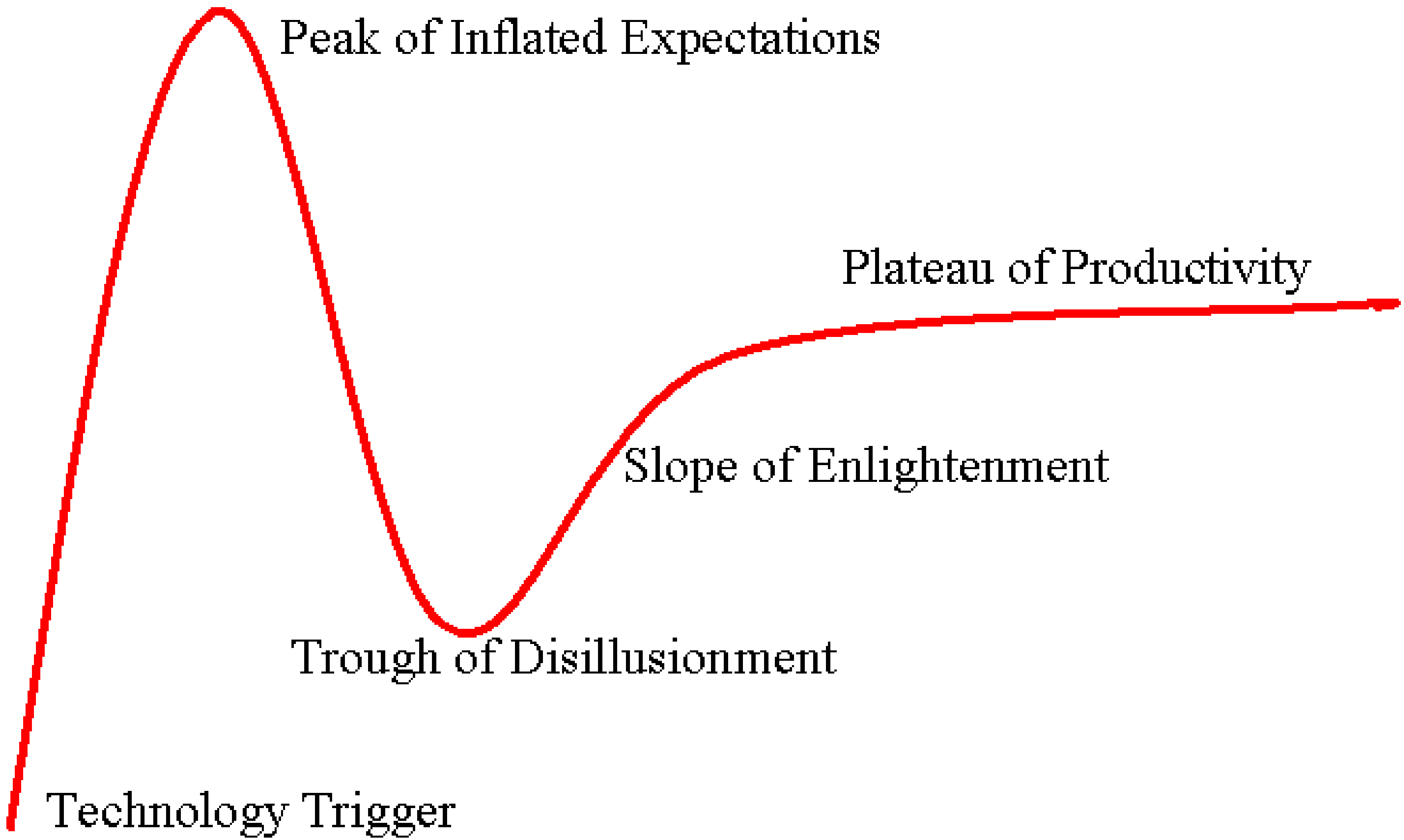
MORGRIDGE INSTITUTE FOR RESEARCH

CENTER FOR HIGH THROUGHPUT COMPUTING

WISCONSIN INSTITUTE FOR DISCOVERY

Solving "real-life" end-to-end problems makes you hype resistance

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!

How do we prepare for the HTC needs of 2020?

Scientific Collaborations at Extreme-Scales:

dV/dt - Accelerating the Rate of Progress towards Extreme Scale Collaborative Science

Collaboration of five institutions – ANL, ISI, UCSD, UND and UW Funded by the *Advanced Scientific Computing Research* (*ASCR*) program of the DOE Office of Science

MORGRIDGE
INSTITUTE FOR RESEARCH

HT
CENTER FOR
HIGH THROUGHPUT
COMPUTING

WISCONSIN
INSTITUTE FOR DISCOVERY

"Using *planning* as the unifying concept for this project, we will develop and evaluate by means of at-scale experimentation novel algorithms and software architectures that will make it less labor intensive for a scientist to **find** the appropriate computing resources, **acquire** those resources, **deploy** the desired applications and data on these resources, and then **manage** them as the applications run. The proposed research will advance the understanding of resource management within a collaboration in the areas of: trust, planning for resource provisioning, and workload, computer, data, and network resource management."

"Over the last 15 years, Condor has evolved from a concept to an essential component of U.S. and international cyberinfrastructure supporting a wide range of research, education, and outreach communities. The Condor team is among the top two or three cyberinfrastructure development teams in the country. In spite of their success, this proposal shows them to be committed to rapid development of new capabilities to assure that Condor remains a competitive offering. Within the NSF portfolio of computational and data-intensive cyberinfrastructure offerings, the High Throughput Computing Condor software system ranks with the NSF High Performance Computing centers in importance for supporting NSF researchers."

A recent anonymous NSF review

"… a mix of continuous changes in technologies, user and application requirements, and the business model of computing capacity acquisition will continue to pose new challenges and opportunities to the effectiveness of scientific HTC. … we have identified six key challenge areas that we believe will drive HTC technologies innovation in the next five years."

- **Evolving resource acquisition models**
- **Hardware complexity**
- **Widely disparate use cases**
- **Data intensive computing**
- **Black-box applications**
- **Scalability**

Thank you HPDC!