Condor Team 2009

Established 1985

Welcome to Condor Week #11

(year #26 for the project)

Main Threads of Activities

- **Distributed Computing Research** – develop and evaluate new concepts, frameworks and technologies that are based on distributed computing principals
- **Software Development & Support** - keep Condor software "**flight worthy**" and support our users
- **The UW Center for High Throughput Computing (CHTC)** – provide HTC capabilities to the UW campus via a distributed computing and storage infrastructure
- **The Open Science Grid (OSG)** – a national, distributed computing partnership for data-intensive research
- **The NSF Middleware Initiative (NMI)** – develop, build and operate a national Build and Test facility

MultiCore **Cyber Infrastructure**

SaaS **HDFS** **WorkFlows**

Workforce **HTPC**

IaaS **Grids**

eResearch

HPC

Cyber Security **100Gb**

SLA

Virtual Machines **QoS**

Web Services

eScience **Open Source**

HTC

Map-Reduce **Clouds** **Master-Worker**

High Availability

Green Computing **GPUs**

Center for Enabling Distributed Petascale Science A SciDAC Center for Enabling Technology

(http://www.cedps.net/images/8/81/CEDPS_ProjectPlan_Oct2009.pdf)

We provide a project plan for the Center for Enabling Distributed Petascale Science (CEDPS), covering the second phase of the project, from October 2009 to June 2011. Our goal in this second phase is to *develop, evaluate, and apply a fully integrated approach to data movement and caching problems that provides an end--to--end solution for extreme-scale data-intensive applications*. This solution uses highly available services, termed **Globus.org**, to provide the needed services; integrates these services into resource managers, such as Condor and the FermiLab glide-in workload management service; and incorporates a state of-the-art logging infrastructure to enhance performance and reliability.

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

TIME *A thinker's guide to the most important trends of the new decade*

In Defense of Failure

By Megan McArdle Thursday, Mar. 11, 2010

"The goal shouldn't be to eliminate failure; it should be to build a system resilient enough to withstand it"

"The real secret of our success is that we learn from the past, and then we forget it. Unfortunately, we're dangerously close to forgetting the most important lessons of our own history: how to fail gracefully and how to get back on our feet with equal grace."

From Forbes Magazine …

Open Source Energy Savings

Dan Woods, 03.02.10, 06:00 AM EST **Software for spreading work over huge collections of computers can be used to cut power costs.**

Condor supports all the operating systems a typical company or research institution would have and is **rock solid** in terms of stability and functions for its intended purpose, which is carving up work and sending it out to any number of computers for processing.

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

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

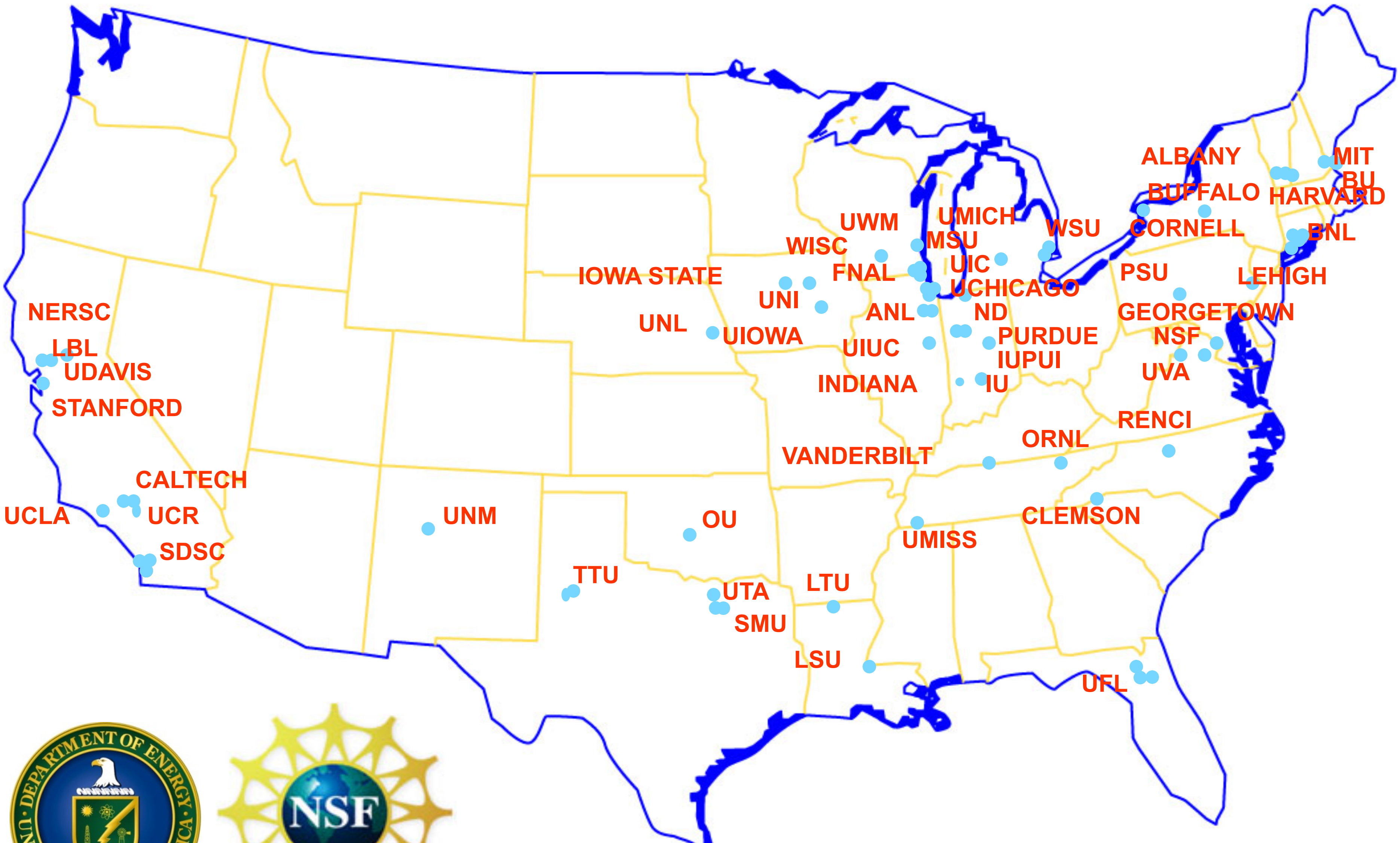
☐ ***UW CAMPUS***

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

Open Science Grid (OSG)



www.cs.wisc.edu/~miron

Native Package in Condor

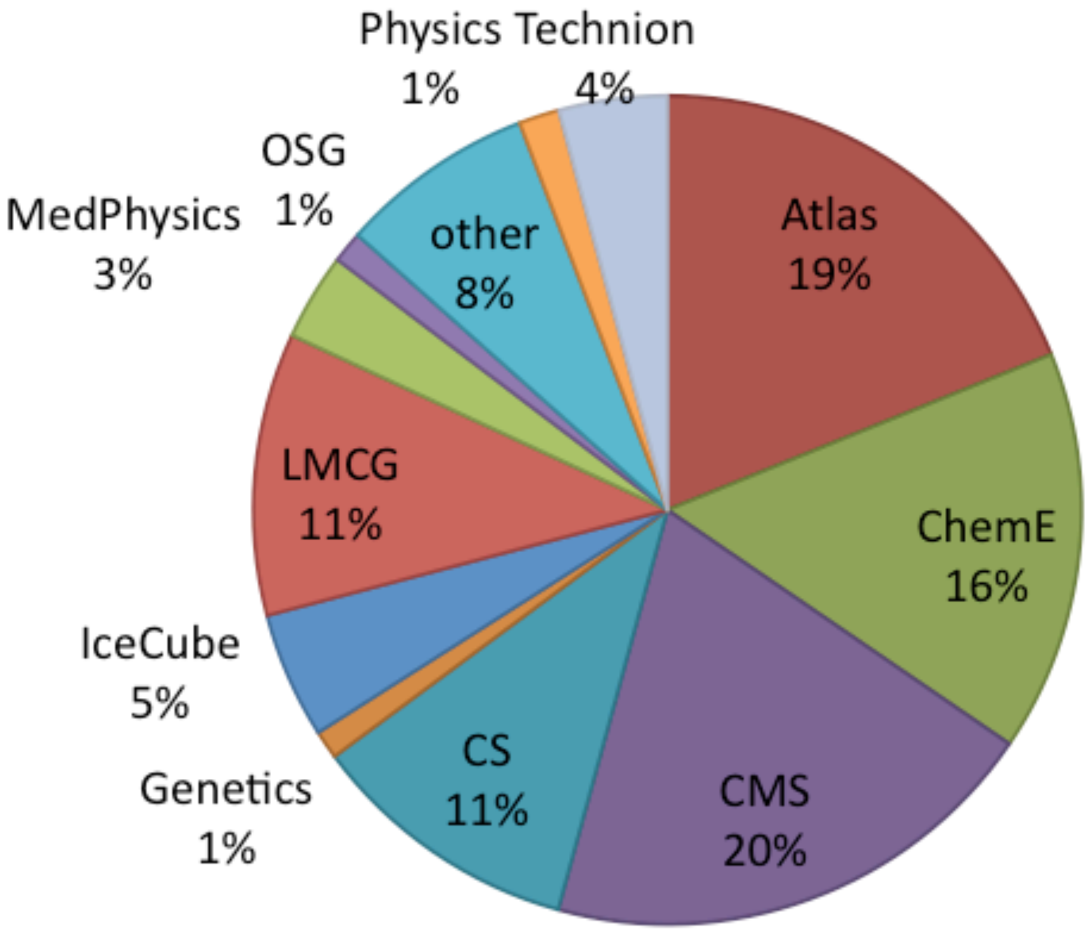
- Beginning in Condor 7.4.x, we have new native packages:
  - Improved RPM with a yum repository
    - Condor "just works" out of the box
    - Condor is now installed in standard locations (/usr/bin, /var/log, …)
    - Creates "condor" user
    - Installs Condor init script
  - New Debian package with deb repository
    - Matches features in RPM (above)
- Condor is included the Fedora Distribution
- Condor is the "G" in REHL MRG
- Condor is a Ubuntu project

HPCwire: Microsoft Injects More Goodies into Windows HPC (Published 09 April 10 02:27 PM)

<…>

For example, Microsoft has added the ability to aggregate Windows 7 workstations into an HPC cluster. Essentially, each workstation is monitored and managed as an ad-hoc compute node within a traditional cluster. Capabilities like time-of-day scheduling and shutting down of preemptive jobs are included so that the machine can be made available to a real live person when required. The common scenario is one where an organization has a small cluster made up of say dozens of servers, along with perhaps hundreds of Windows 7 PCs sitting idle at night.

GLOW Usage 04/04-04/10



114M Hours

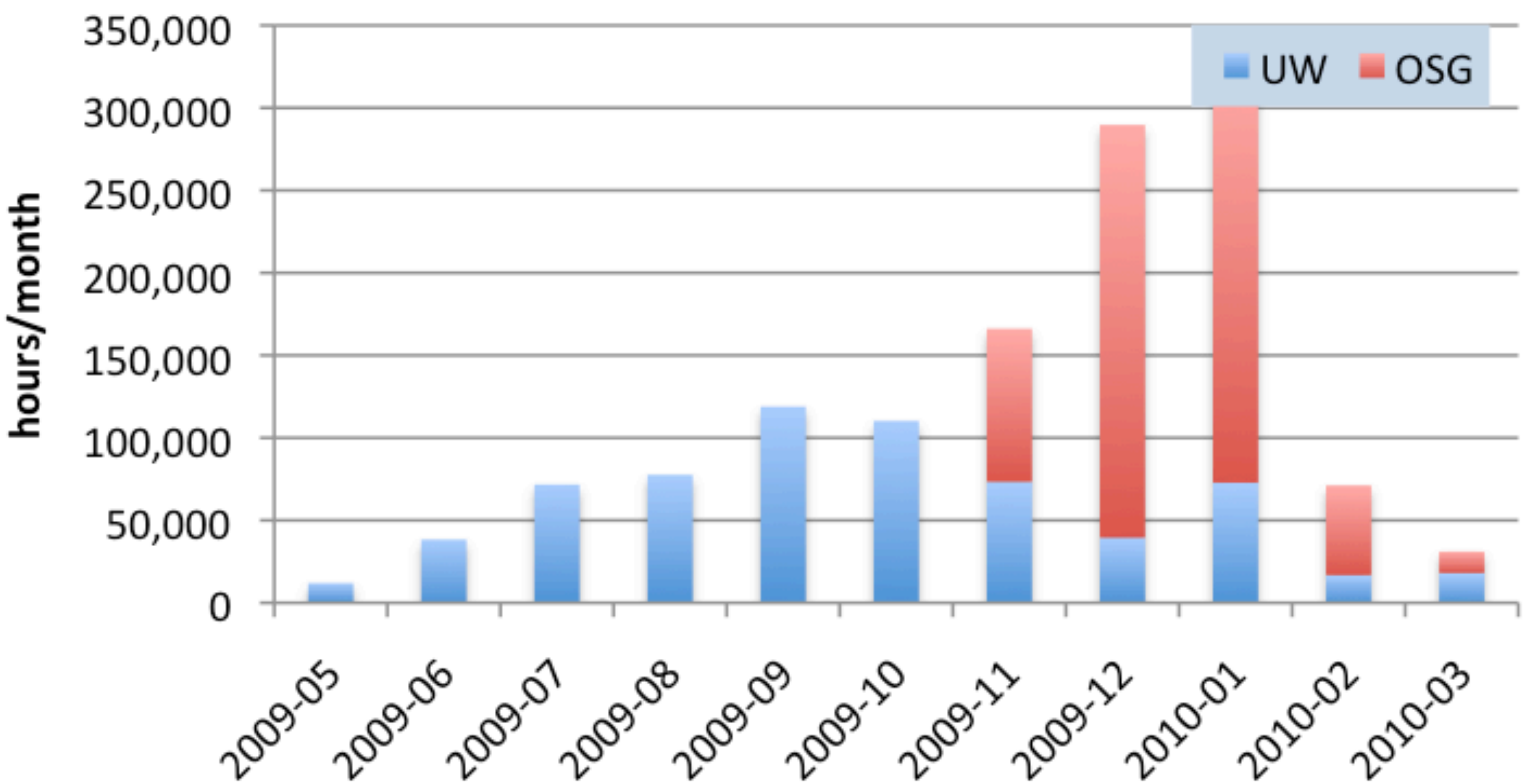
The **Qiang Cui (QC) Group**

"We develop and apply theoretical tools (electronic structure, nuclear dynamics and statistical mechanics) to biophysical problems (enzyme catalysis, bioenergy transduction and biomaterials etc.)."

http://kandinsky.chem.wisc.edu/~qiang/

Started to use the High Through Parallel Computing (HTPC) provided by CHTC in 05/09 and expended to OSG in 11/09

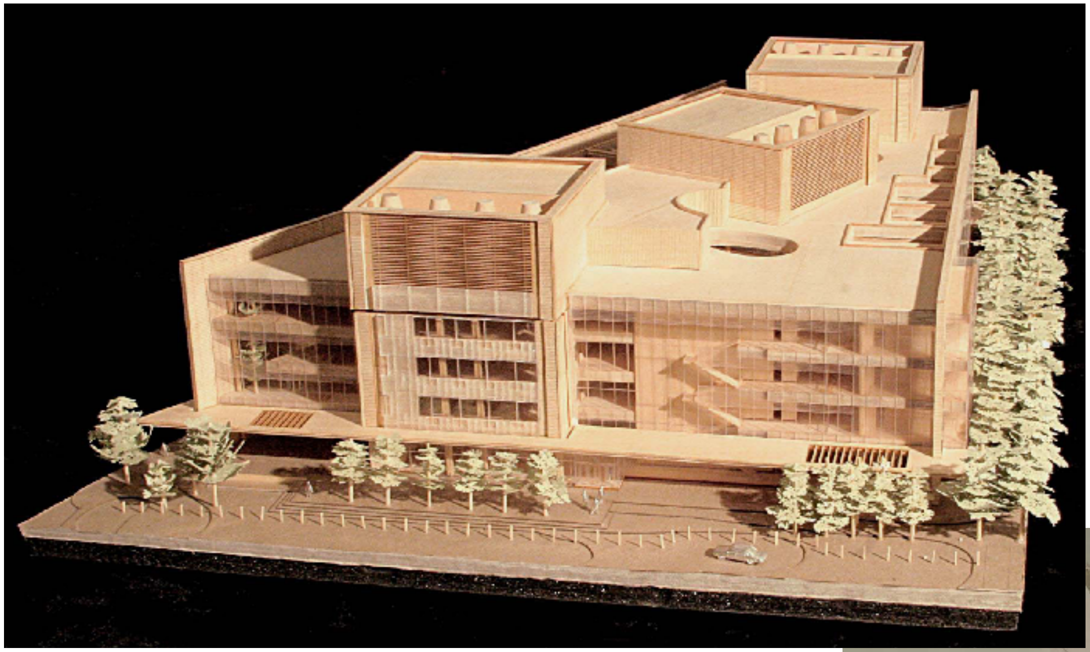
QC Group Comsumption



Impact of these cycles

1. Iron-Catalyzed Oxidation Intermediates Captured in A DNA Repair Monooxygenase, C. Yi, G. Jia, G. Hou, Q. Dai, G. Zheng, X. Jian, C. G. Yang, Q. Cui, and C. He, **Science**, Submitted
2. Disruption and formation of surface salt bridges are coupled to DNA binding in integration host factor (IHF): a computational analysis, L. Ma, M. T. Record, Jr., N. Sundlass, R. T. Raines and Q. Cui**, J. Mol. Biol,** Submitted
3. An implicit solvent model for SCC-DFTB with Charge-Dependent Radii, G. Hou, X. Zhu and Q. Cui**, J. Chem. Theo. Comp.**, Submitted
4. Sequence-dependent interaction of β-peptides with membranes, J. Mondal, X. Zhu, Q. Cui and A. Yethiraj, **J. Am. Chem. Soc**., Submitted
5. A new coarse-grained model for water: The importance of electrostatic interactions, Z. Wu, Q. Cui and A. Yethiraj**, J. Phys. Chem**. Submitted
6. How does bone sialoprotein promote the nucleation of hydroxyapatite? A molecular dynamics study using model peptides of different conformations, Y. Yang, Q. Cui, and N. Sahai, {\it Langmuir}, Submitted
7. Preferential interactions between small solutes and the protein backbone: A computational analysis, L. Ma, L. Pegram, M. T. Record, Jr., Q. Cui, **Biochem**., 49, 1954-1962 (2010)
8. Establishing effective simulation protocols for β- and α/β-peptides. III. Molecular Mechanical (MM) model for a non-cyclic β-residue, X. Zhu, P. K\"onig, M. Hoffman, A. Yethiraj and Q. Cui**, J. Comp. Chem.**, In press (DOI: 10.1002/jcc.21493)
9. Curvature Generation and Pressure Profile in Membrane with lysolipids: Insights from coarse-grained simulations, J. Yoo and Q. Cui, **Biophys. J**. 97, 2267-2276 (2009)

Two new Institutes on the UW Campus - MIR & WID



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

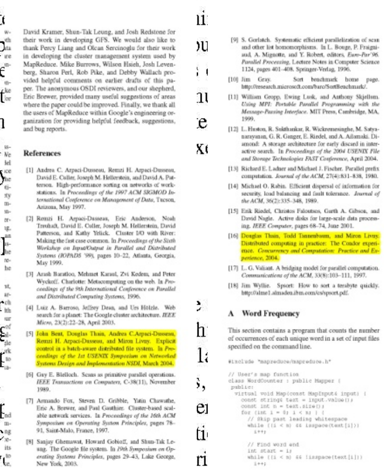
Abstract

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.

Ou … M … large
clust … ii … able:
a typ … ou … y ter-
abyte … mers
find t … e pro-
gram … e thou-
sand … xe … isters
every

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].



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++;
```

1

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

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
meas

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.

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

LIGO

From: Stuart Anderson <anderson@ligo.caltech.edu>
Date: February 28, 2010 11:51:32 PM EST
To: Condor-LIGO mailing list <condorligo@aei.mpg.de>
Subject: [CondorLIGO] Largest LIGO workflow

Pete,
Here are some numbers you ask about for LIGO's use of DAGs to manage large data analysis tasks broken down by the largest number of jobs managed in different categories:

1) DAG Instance--one condor_dagman process: **196,862**.
2) DAG Workflow--launched from a single condor_submit_dag but may include multiple automatic sub- or spliced DAGs: **1,120,659**.
3) DAG Analysis--multiple instances of condor_submit_dag to analyze a common dataset with results combined into a single coherent scientific result: **6,200,000**.
4) DAG Total--sum over all instances of condor dagman run: **O (100M)**.

P.S. These are lower bounds as I did not perform an exhaustive survey/ search, but they are probably close.

Thanks.

Thank you for building such

a wonderful community