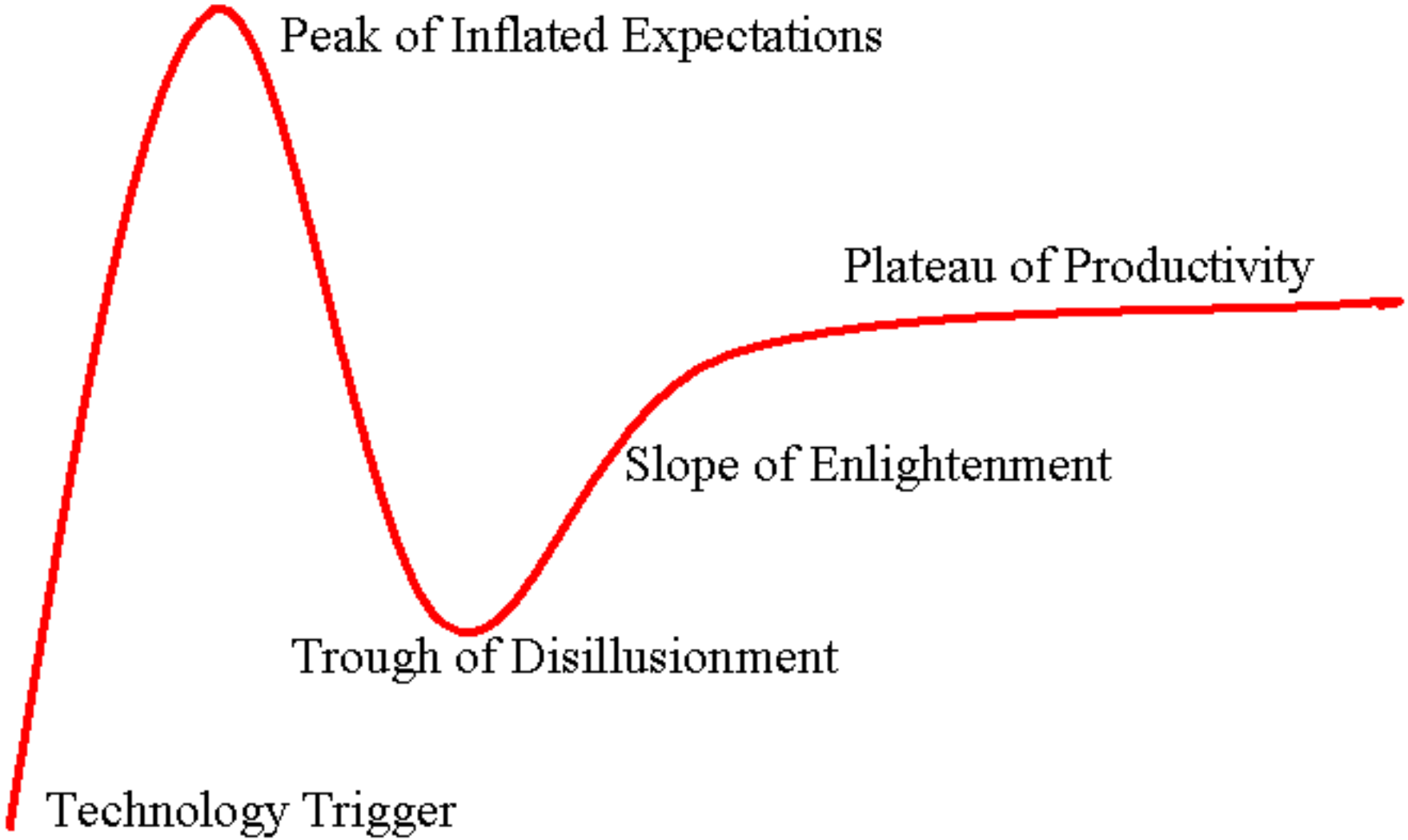
From Distributed Processing Systems to the buzz word of the day and back

Miron Livny
Wisconsin Institutes for Discovery
Madison-Wisconsin

While I do not know (or understand) your problem, I am sure that my XYZ software will solve it!

Gartner Hype Cycle



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

Ecclesiastes Chapter 1 verse 9

Perspectives on Grid Computing

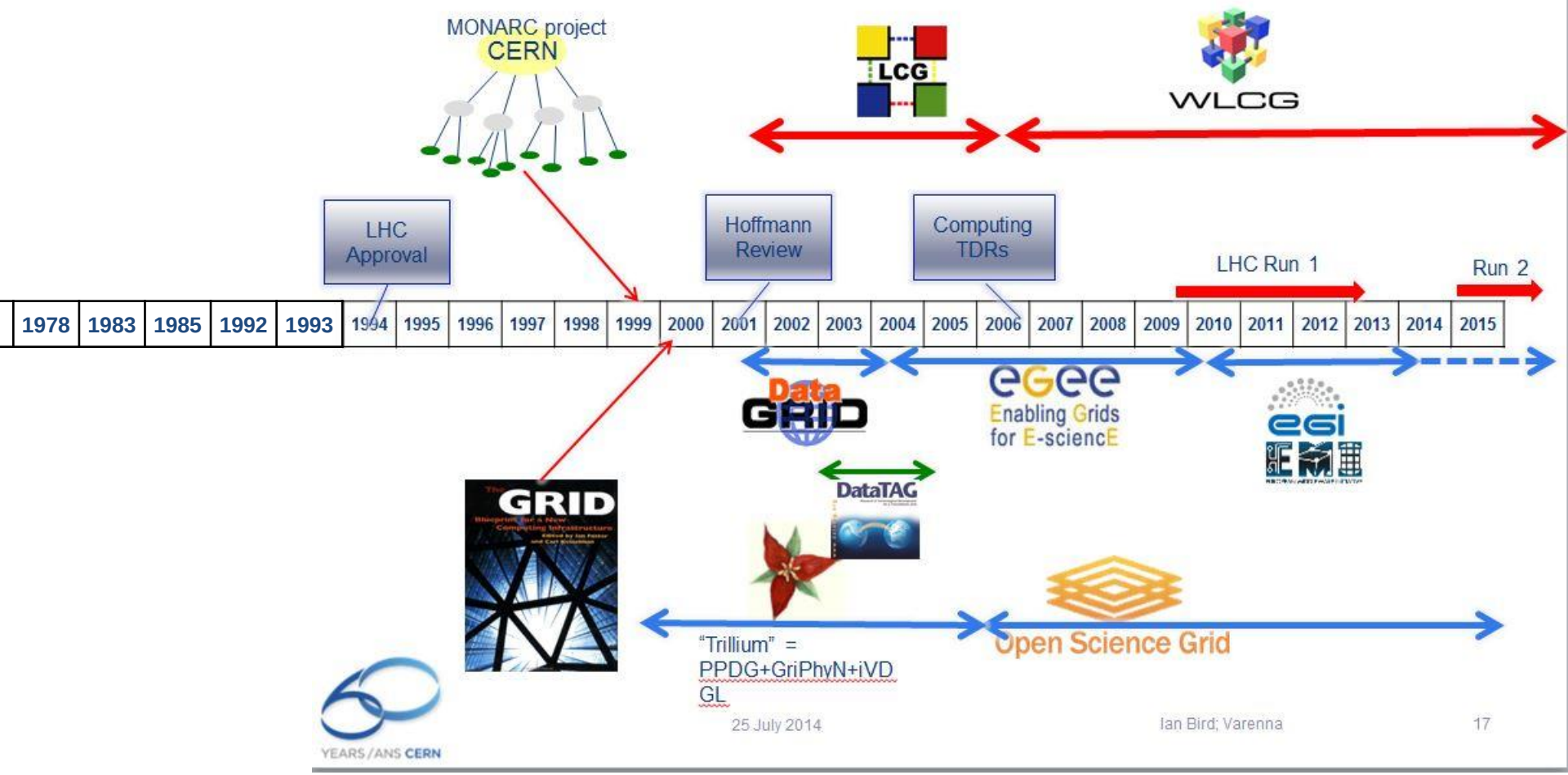
Uwe Schwiegelshohn Rosa M. Badia Marian Bubak Marco Danelutto
Schahram Dustdar Fabrizio Gagliardi Alfred Geiger Ladislav Hluchy
Dieter Kranzlmüller Erwin Laure Thierry Priol Alexander Reinefeld
Michael Resch Andreas Reuter Otto Rienhoff Thomas Rüter Peter Sloot Domenico
Talia Klaus Ullmann Ramin Yahyapour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

HTCondor Team 2013

Project Established 1985

Timeline of projects



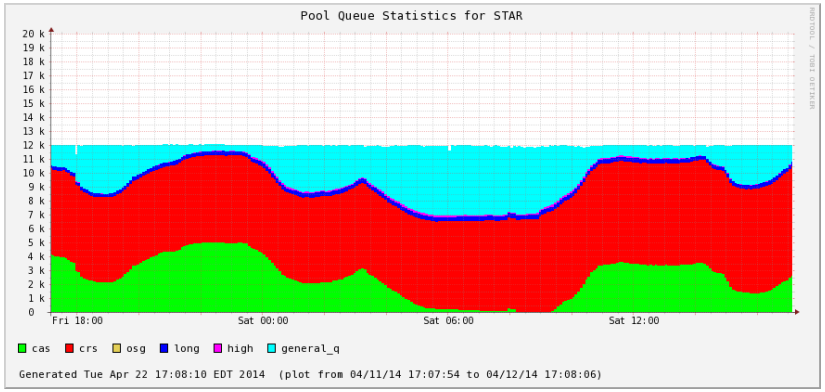
“Over the last 15 years, Condor has evolved from a concept to an essential component of U.S. and international cyberinfrastructure supporting a wide range of research, education, and outreach communities. The Condor team is among the top two or three cyberinfrastructure development teams in the country. 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.”

An anonymous NSF review (04/2013)

RACF Overview

Main HTCondor pools

- PHENIX—12.2kCPU
- STAR—12.0kCPU
- ATLAS—13.1kCPU



STAR/PHENIX are RHIC detectors

- Loose federation of individual users

ATLAS—tightly controlled, subordinate to PANDA workflow management, strict structure

Smaller Experiments

- LBNE
- Dayabay
- LSST

Migration to HTCondor

- Torque/Maui had been used for many years
  - Many issues
  - Severity & number of problems increased as size of farm increased
- Migration

2012 Aug	Started evaluating alternatives to Torque/Maui
	(LSF, Grid Engine, Torque 4, HTCondor, SLURM)
2013 Jun	Began testing HTCondor with ATLAS & CMS
2013 Aug	Choice of HTCondor approved by management
2013 Sep	HTCondor declared production service
	Moved 50% of pledged CPU resources to HTCondor
2013 Nov	Migrated remaining resources to HTCondor

Services (2)

Batch:

- SLC6 migration: SLC5 CEs decommissioned, no grid job submission to SLC5
  - SLC5 WNs final migration ongoing
- Batch system migration, from LSF to HTCondor
  - Goals: scalability, dynamism, dispatch rate, query scaling
  - Replacement candidates:
    - SLURM feels too young
    - HTCondor mature and promising
    - Son of Grid Engine fast, a bit rough
  - More details of selection process: https://indico.cern.ch/event/247864/session/5/contribution/22/material/slides/0.pdf

It is all (mainly) about automation

"Do what I told you to do and let me know when you are done!"

Automation requires dependable, capable and affordable mechanisms that are controlled by software that implements the policies of the end user. This is why mechanisms determine what a computing system can do for you and it is much harder to change/improve mechanisms then policies.

We prefer to talk and do policies. However, it is critical that we talk and understand mechanisms. This requires an understanding of principals and problems not solutions.

Computer Science problems do not die

The good (bad) news about these problems is that they have many non-optimal solutions as they are all based on tradeoffs that cannot be (easily) quantified and are affected by frequent changes in how computers are used and the technologies (hardware and software) that is used to implement/build them.

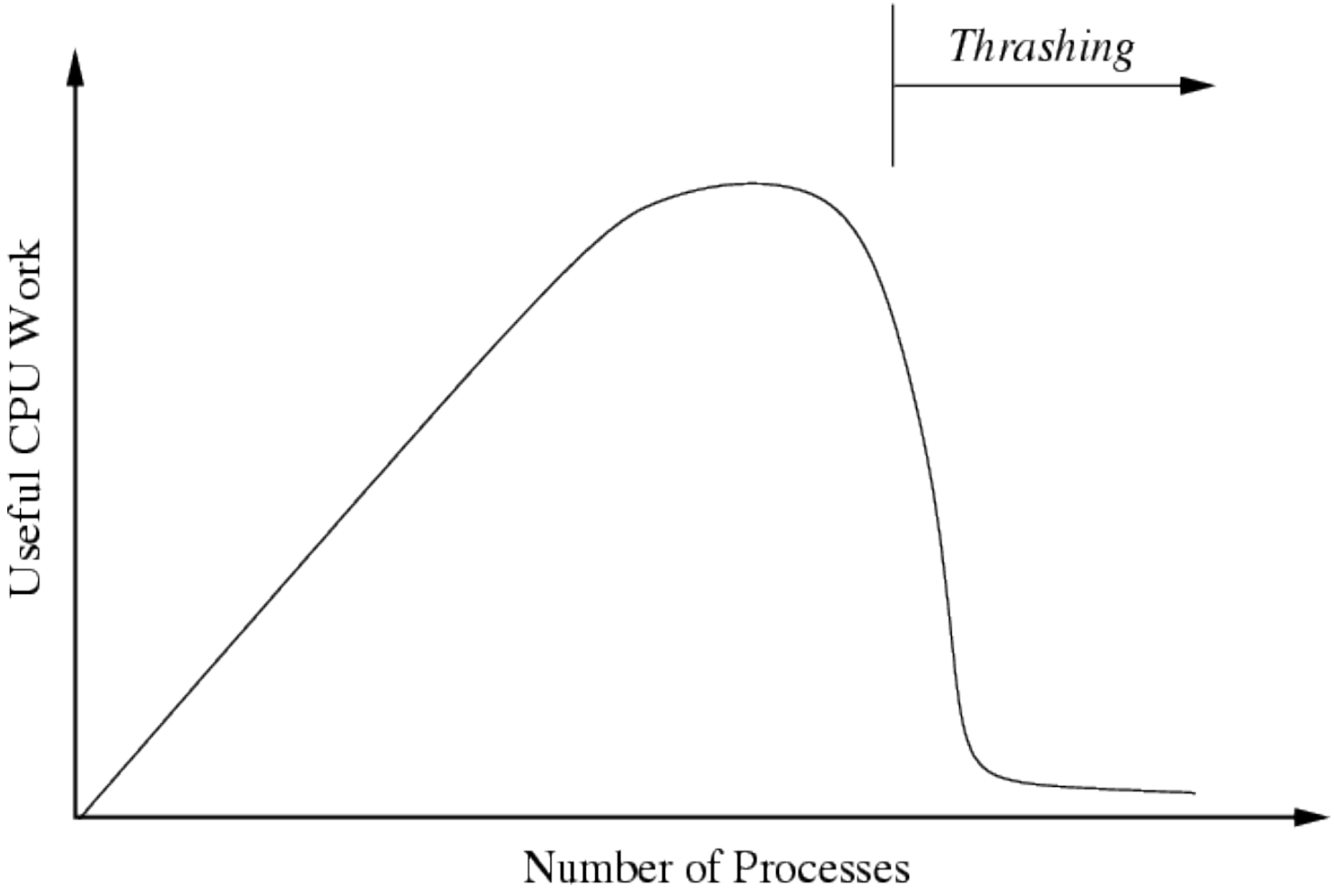
Problems that do not go way

- Name spaces – when you say “give me foo” how do I know what “foo” is?
- Caching – what should I remove when the cache is full?
- Co-scheduling – which resource should I hold while I am waiting for to other resource to become available?
- Error propagation – how do I tell you that I failed?
- Verification of a computing system – How do I know that the system does what it is supposed to do?
- Checkpoint restart of applications – How much would it cost to allocate the resource you are using now to someone else?

Multiprogramming

Maximize throughput by overlapping CPU and I/O, can lead to thrashing due to virtual memory contention (poor locality of reference) and therefore requires inter-application protection.

Maximize utilization of all resources. Individual running time is not the focus (objective).


Thrashing
Useful CPU Work
Number of Processes

What is fair?

- How long should I wait?
- How much should I pay?
- How fast will I run?
- How predictive will the service be?
- How will I know that I am treated fairly?
- How will I know that my resources are allocated fairly?

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

The paradigm shift of 70's – computing hardware sold in small units.

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**

Multiplicity of resources

The system should provide a number of assignable resources for any type of **service** demand. The greater the degree of replication of resources, the better the ability of the system to maintain high reliability and performance

Component interconnection

A Distributed System should include a communication subnet which interconnects the elements of the system. The transfer of information via the subnet should be controlled by a two-party, cooperative protocol (**loose coupling**).

System transparency

From the users point of view the set of resources that constitutes the Distributed Processing System acts like a "**single virtual machine**". When **requesting a service** the user should not require to be aware of the physical location or the instantaneous load of the various resources

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.

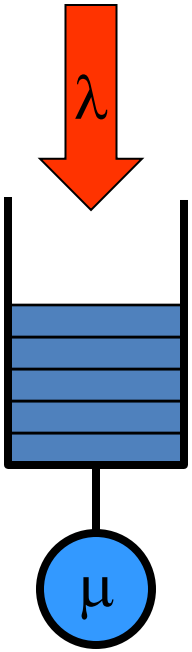
Challenges

- Race Conditions…
- Name spaces …
- Distributed ownership …
- Heterogeneity …
- Object addressing …
- Data caching …
- Object Identity …
- Trouble shooting …
- Circuit breakers …

One centralized queue or many distributed queues?

Or, the Wait while Idle problem.

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

M/M/2 – One queue 2 servers

Utilization

Number Of Customers In the Queue

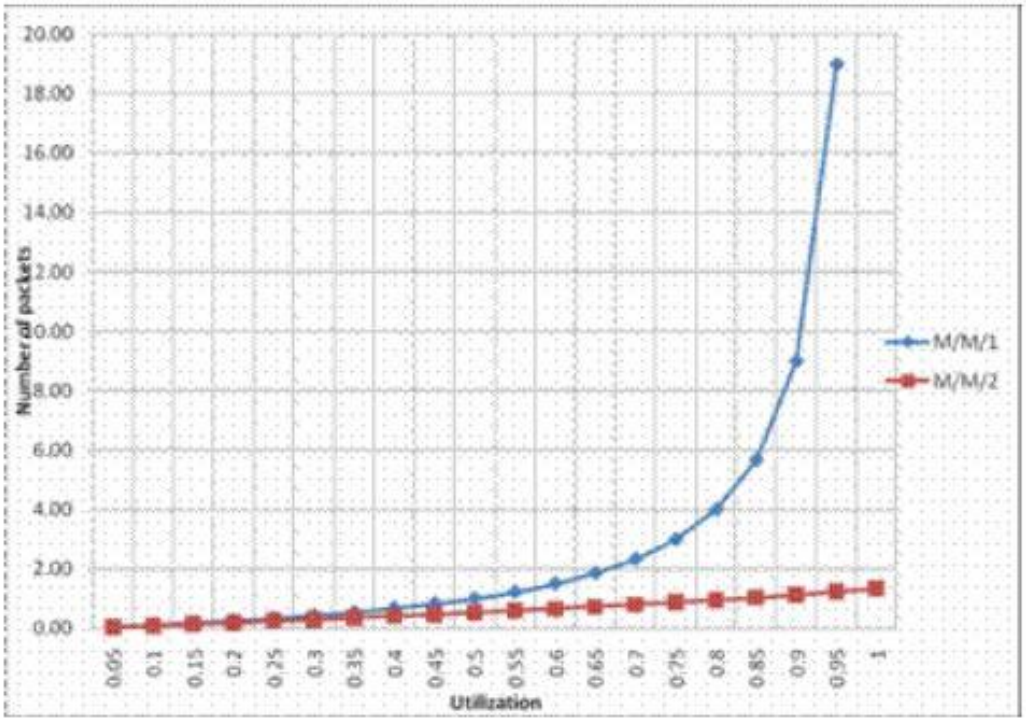
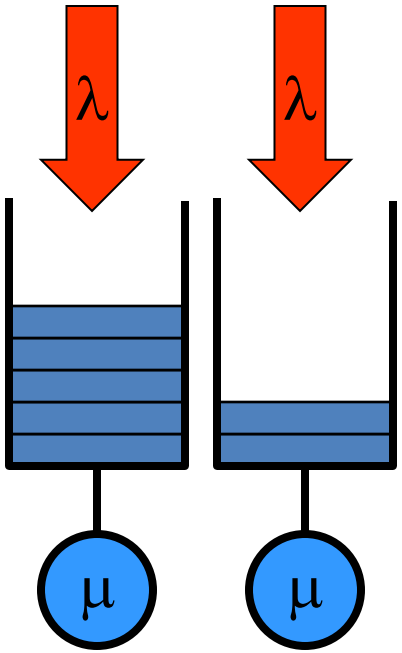


Figure 8. *Mean number in System versus utilization for M/M/1 and M/M/2 analytical queue models*

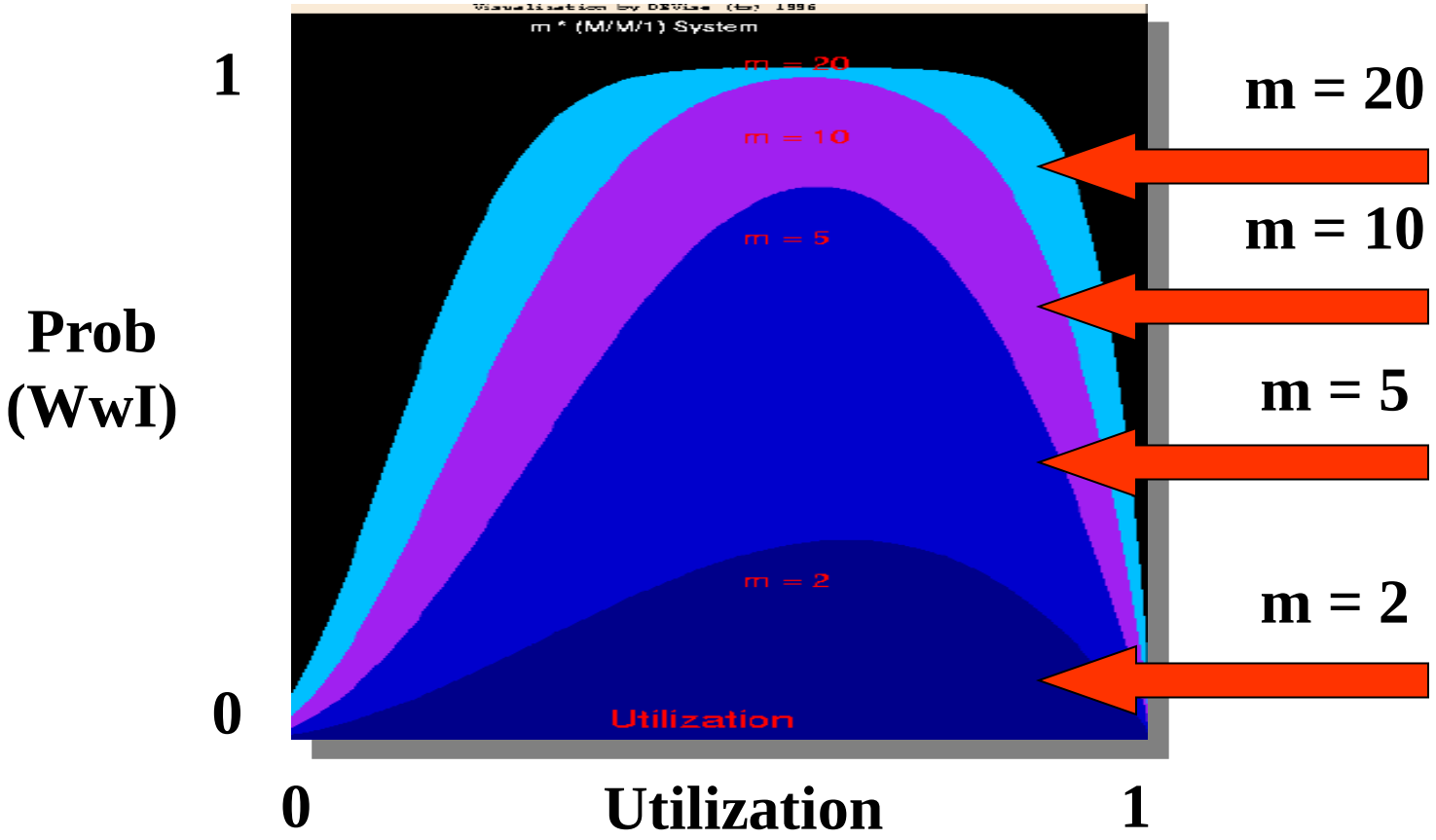
What about 2*M/M/1 ?



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 and wait or should I move to another queue?

" … 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 1984 we introduced the concept of "distributed ownership", developed our first checkpointing capability and started to implement the first version of Condor leveraging a remote I/O capability (split execution) that was developed at our CS department in 1982.

First version of Condor was installed on 20 DEC2 (desk top) workstations to serve our CS department in 1985.

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

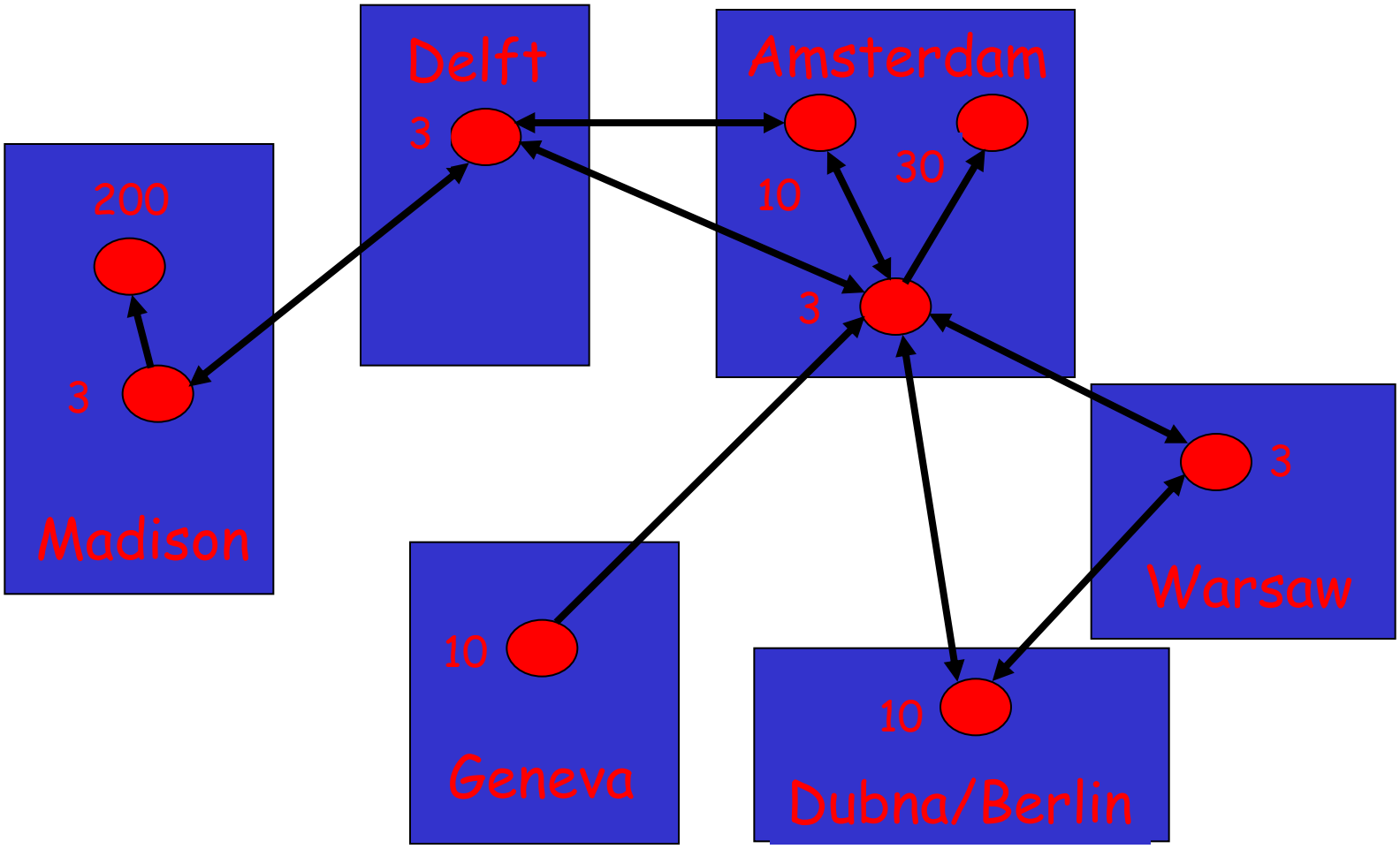
Use wide-area networks to transfer batch jobs between Condor systems

- 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



D. H. J Epema, Miron Livny, R. van Dantzig, X. Evers, and Jim Pruyne, "A Worldwide Flock of Condors : Load Sharing among Workstation Clusters" *Journal on Future Generations of Computer Systems*, Volume 12, 1996

Use resource and job management "gateways" to connect the Condor pools.

Established a Peer to Peer relationship between the pools to support full local control.

Followed the routing approach of message passing networks to establish a connection between the source (owner of the work) and the destination (resource).

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**.

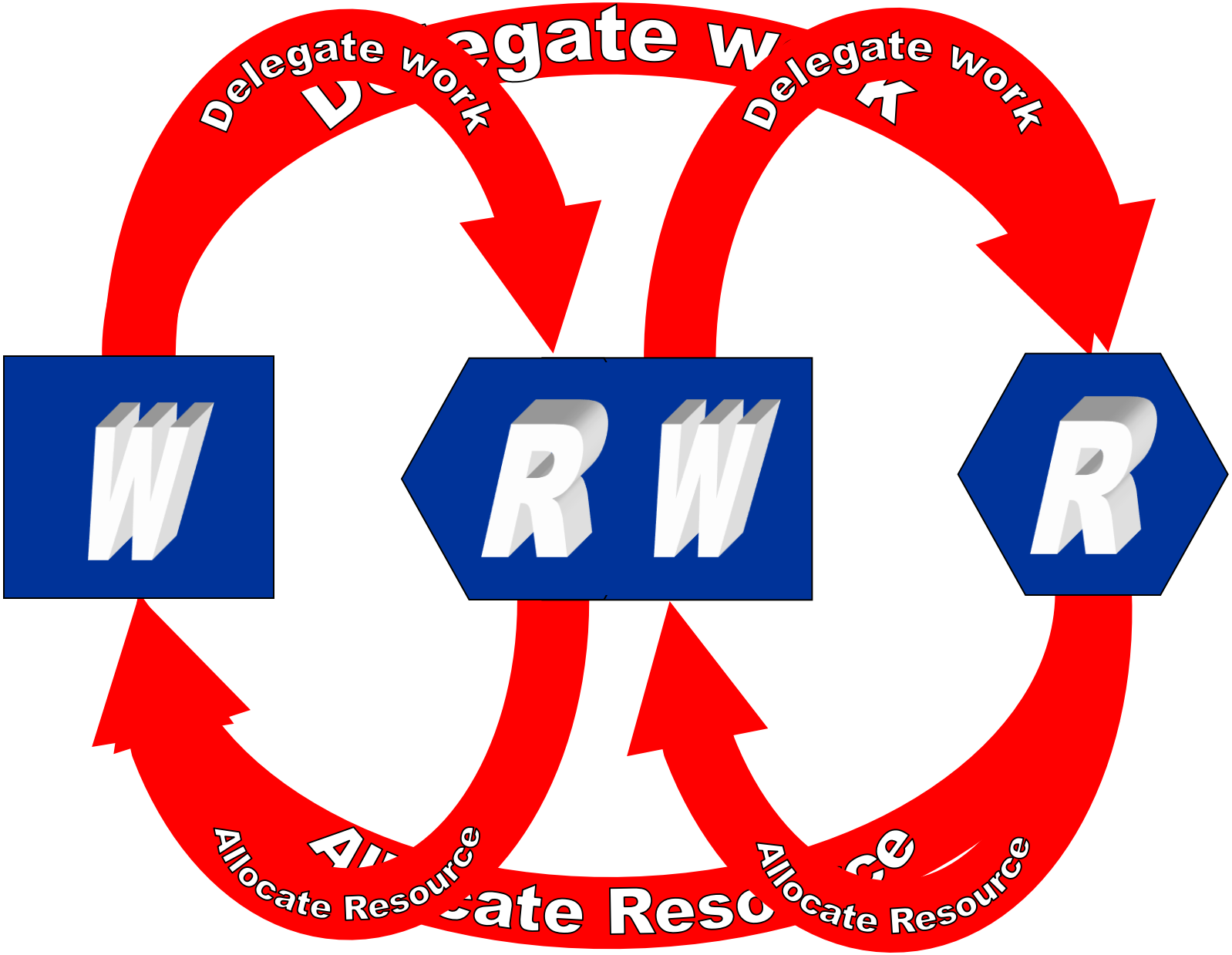
High Throughput Computing is a 24-7-365 activity and therefore requires automation

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)

Resource Allocation
(resource -> job)
vs.
Work Delegation
(job -> resource)


Delegate work
Delegate work
W
R W
R
Allocate Resource
Allocate Resource

Resource Allocation

A limited assignment of the "ownership" of a resource

- Owner is charged for allocation regardless of actual consumption
- Owner can allocate resource to others
- Owner has the right and means to revoke an allocation
- Allocation is governed by an "agreement" between the client and the owner
- Allocation is a "lease"
- Tree of allocations

"We present some principles that we believe should apply in any compute resource management system. The first, P1, speaks to the need to avoid "resource leaks" of all kinds, as might result, for example, from a monitoring system that consumes a nontrivial number of resources.

P1 - It must be possible to monitor and control *all* resources consumed by a CE—whether for "computation" or "management."

Our second principle is a corollary of P1:

P2 - A system should incorporate circuit breakers to protect both the compute resource and clients. For example, negotiating with a CE consumes resources. How do we prevent an eager client from turning into a denial of service attack? "

Ian Foster & Miron Livny, *"Virtualization and Management of Compute Resources: Principles and Architecture ", A working document (February 2005)*

Garbage collection is the cornerstone of resource allocation

Work Delegation

A limited assignment of the responsibility to perform the work

- Delegation involved a definition of these "responsibilities"
- Responsibilities my be further delegated
- Delegation consumes resources
- Delegation is a "lease"
- Tree of delegations

HTCondor Job Submission Options

- leave_in_queue = <ClassAd Boolean Expression>
- on_exit_remove = <ClassAd Boolean Expression>
- on_exit_hold = <ClassAd Boolean Expression>
- periodic_remove = <ClassAd Boolean Expression>
- periodic_hold = <ClassAd Boolean Expression>
- periodic_release = <ClassAd Boolean Expression>
- noop_job = <ClassAd Boolean Expression>

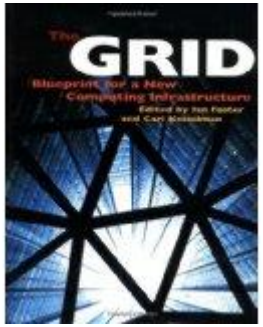
The Grid Movement

Enable (limited) work delegation and remote (distributed) authorization that is based on a global identity namespace and a (small) group of trusted certificate authorities.

Introduced authentication to Distributed Processing Systems.

The Grid: Blueprint for a New Computing Infrastructure

Edited by Ian Foster and Carl Kesselman
July 1998, 701 pages.



The grid promises to fundamentally change the way we think about and use computing. This infrastructure will connect multiple regional and national computational grids, creating a universal source of **pervasive and dependable** computing power that supports dramatically new classes of applications. The Grid provides a clear vision of what computational **grids** are, why we need them, who will use them, and how they will be programmed.

" … 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*".**

" … Grid computing is a partnership between clients and servers. Grid clients have more responsibilities than traditional clients, and must be equipped with powerful mechanisms for dealing with and recovering from failures, whether they occur in the context of remote execution, work management, or data output. When clients are powerful, servers must accommodate them by using careful protocols.… "

Douglas Thain & Miron Livny, *"Building Reliable Clients and Servers", in "The Grid: Blueprint for a New Computing Infrastructure", 2nd edition*

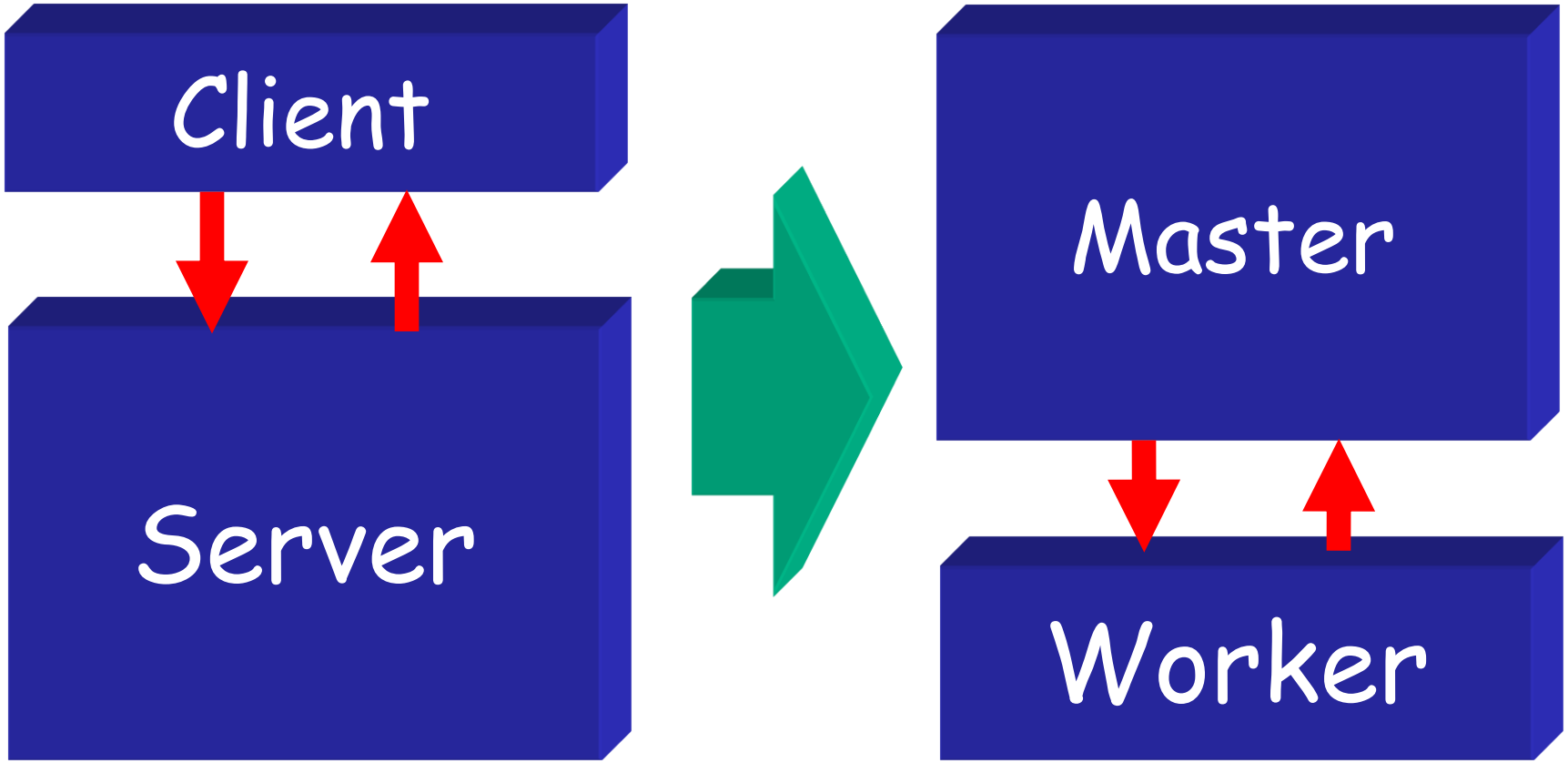
The Ethernet Protocol

IEEE 802.3 CSMA/CD - A truly distributed (and very effective) access control protocol to a shared service.

- Client responsible for access control
- Client responsible for error detection
- Client responsible for fairness

WWW

Grid



The NUG30 Quadratic Assignment Problem (QAP)

$$\min_{p \in \Pi} \sum_{i=1}^{30} \sum_{j=1}^{30} a_{ij} b_{p(i)p(j)}$$

Solved!

(4 Scientists + 1 Linux Box)

NUG30 Personal Grid (06/2000)

Managed by **one** Linux box at Wisconsin

Flocking:
-- the main Condor pool at Wisconsin (500 processors)
-- the Condor pool at Georgia Tech (284 Linux boxes)
-- the Condor pool at UNM (40 processors)
-- the Condor pool at Columbia (16 processors)
-- the Condor pool at Northwestern (12 processors)
-- the Condor pool at NCSA (65 processors)
-- the Condor pool at INFN Italy (54 processors)

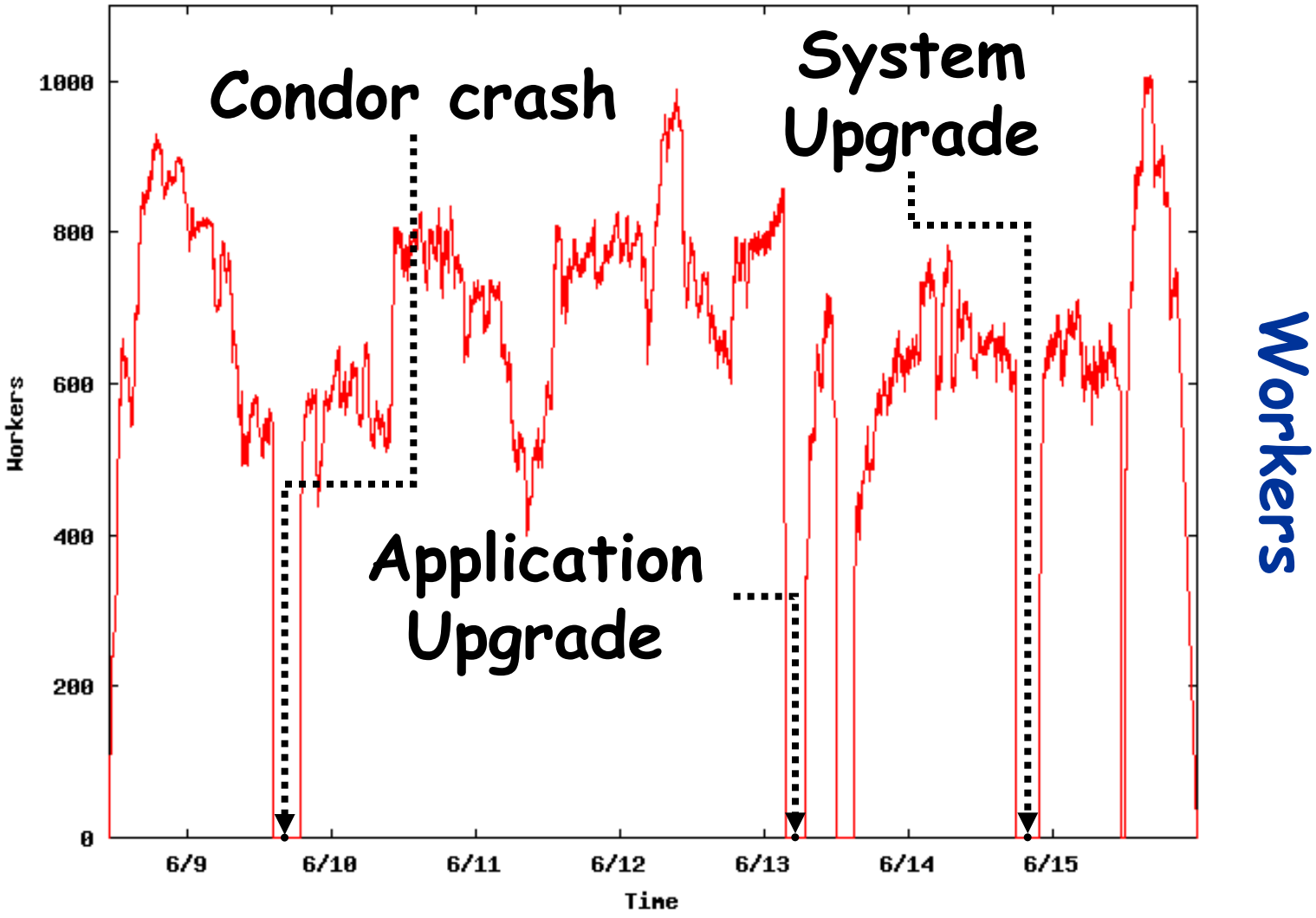
Glide-in:
-- Origin 2000 (through LSF) at NCSA. (512 processors)
-- Origin 2000 (through LSF) at Argonne (96 processors)

Hobble-in:
-- Chiba City Linux cluster (through PBS) at Argonne (414 processors).

Solution Characteristics.

Scientists	**4**
Workstations	**1**
Wall Clock Time	6:22:04:31
Avg. # CPUs	653
Max. # CPUs	1007
Total CPU Time	Approx. 11 years
Nodes	11,892,208,412
LAPs	574,254,156,532
Parallel Efficiency	92%

The NUG30 Workforce



Being a Master

Customer "delegates" task(s) to the master who is responsible for:

- Obtaining **allocation** of resources
- Deploying and managing workers on allocated resources
- **Delegating** work unites to deployed workers
- Receiving and processing results
- Delivering results to customer

Master must be …

- Persistent – work and results must be safely recorded on non-volatile media
- Resourceful – delegates "DAGs" of work to other masters
- Speculative – takes chances and knows how to recover from failure
- Self aware – knows its own capabilities and limitations
- Obedience – manages work according to plan
- Reliable – can mange "large" numbers of work items and resource providers
- Portable – can be deployed "on the fly" to act as a "sub master"

Master should not do …

- Predictions …
- Optimal scheduling …
- Data mining …
- Bidding …
- Forecasting …

Dear Master,

Never assume that *what you know* is still true and that *what you ordered* did actually happen!

Every Community can benefit from the services of

Matchmakers!

eBay is a matchmaker

Why? Because ...

.. someone has to bring together community members who have requests for goods and services with members who offer them.

- Both sides are looking for each other
- Both sides have constraints
- Both sides have preferences

Being a Matchmaker

- Symmetric treatment of all parties
- Schema "neutral"
- Matching policies defined by parties
- "Just in time" decisions
- Acts as an "advisor" not "enforcer"
- Can be used for "resource allocation" and "job delegation"

The paradigm shift of 00's – computing capacity sold on demand for short time periods.

In other words, computing capacity is assumed to be unbounded as long as you have an unbounded CC.

rence started!

s and 12 hours

List of Panels

- Declarative, Domain-Specific Languages - Elegant Simplicity or a Hammer in S of a Nail?
  Sam Madden, *MIT*; Alan Demers, *Cornell University*; Michael Carey, *BEA*; Boon Loo, *University of Pennsylvania*; John Whaley, *moka5*
- Scientific Data Management: An Orphan in the Database Community?
  Randal Burns (moderator), *Johns Hopkins University*; Susan B. Davidson, *Corn University*; Yannis Ioannidis, *University of Athens*; Miron Livny, *University of Wisconsin-Madison*; Jignesh M. Patel, *University of Michigan*
- Cloud Computing-Was Thomas Watson Right After All?
  Raghu Ramakrishnan (moderator), *Yahoo!*; Eric Baldeschwieler, *Yahoo!*; James Hamilton, *Microsoft*; Miron Livny, *University Wisconsin-Madison*; Yossi Matias, *Google*; Hamid Pirahesh, *IBM*

Dates

Condor in the Clouds

rence started!

s and 12 hours

List of Panels

o

Dates

s

n

ommittee

on Committee

- Declarative, Domain-Specific Languages - Elegant Simplicity or a Hammer in S of a Nail?
  Sam Madden, *MIT*; Alan Demers, *Cornell University*; Michael Carey, *BEA*; Boon Loo, *University of Pennsylvania*; John Whaley, *moka5*
- Scientific Data Management: An Orphan in the Database Community?
  Randal Burns (moderator), *Johns Hopkins University*; Susan B. Davidson, *Corn University*; Yannis Ioannidis, *University of Athens*; Miron Livny, *University of Wisconsin-Madison*; Jignesh M. Patel, *University of Michigan*
- Cloud Computing-Was Thomas Watson Right After All?
  Raghu Ramakrishnan (moderator), *Yahoo!*; Eric Baldeschwieler, *Yahoo!*; James Hamilton, *Microsoft*; Miron Livny, *University Wisconsin-Madison*; Yossi Matias, *Google*; Hamid Pirahesh, *IBM*

CONDOR high throughput computing

THE UNIVERSITY of WISCONSIN MADISON

The words of Koheleth son of David, king in Jerusalem . . .

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

rence started!

s and 12 hours

List of Panels

fo

Dates

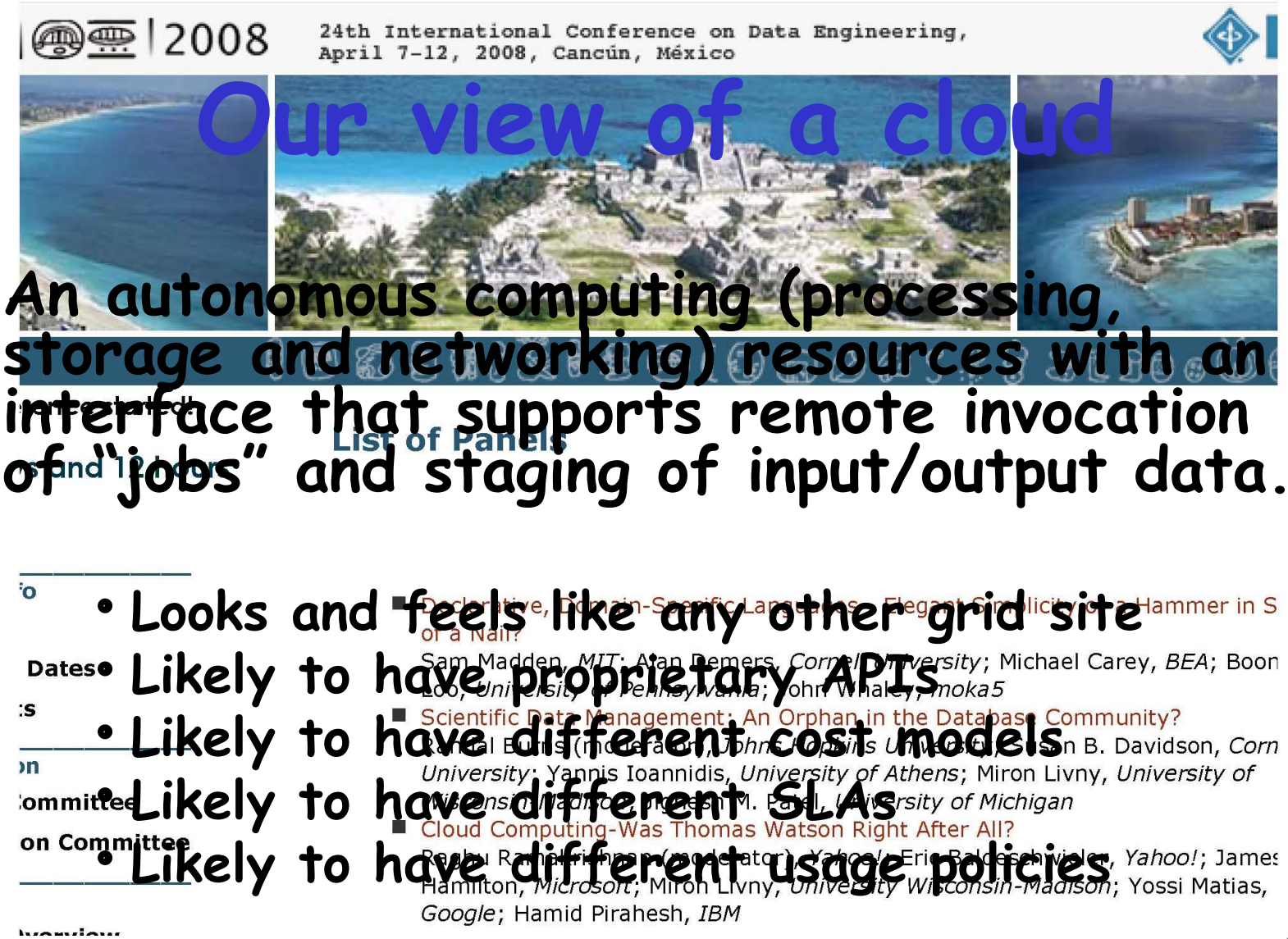
ts

on

ommittee

on Committee

- Declarative, Domain-Specific Languages - Elegant Simplicity or a Hammer in S of Mud?
  Sam Madden, *MIT*; Alan Demers, *Cornell University*; Michael Carey, *BEA*; Boon Loo, *University of Pennsylvania*; John Whaley, *moka5*
- Scientific Data Management: An Orphan in the Database Community?
  Randal Burns (moderator), *Johns Hopkins University*; Susan B. Davidson, *Corn University*; Yannis Ioannidis, *University of Athens*; Miron Livny, *University of Wisconsin-Madison*; Jignesh M. Patel, *University of Michigan*
- Cloud Computing-Was Thomas Watson Right After All?
  Raghu Ramakrishnan (moderator), *Yahoo!*; Eric Baldeschwieler, *Yahoo!*; James Hamilton, *Microsoft*; Miron Livny, *University Wisconsin-Madison*; Yossi Matias, *Google*; Hamid Pirahesh, *IBM*



Our view of a cloud

An autonomous computing (processing, storage and networking) resources with an interface that supports remote invocation of "jobs" and staging of input/output data.

- Looks and feels like any other grid site
- Likely to have proprietary APIs
- Likely to have different cost models
- Likely to have different SLAs
- Likely to have different usage policies

What do we do with clouds?

- Turn VMs into "first class citizens" in the Condor framework
- Interact with users in academia and industry who express interest in using computing resources offered by clouds
- Add EC2+S3 to the (long) list of remote resources Condor can harness (or delegate work to)
- Explore possible enhancements to our matchmaking and workflow technologies to support provisioning of cloud resources (including inter-cloud migration)
- Understand the semantics of the EC2+S3 services, protocols and infrastructure so that we can provide a Condor "overlay" that expend local capabilities to include these resources
- Monitor new cloud "formations"

How can I use Condor?

- As a job manager and resource scheduler for a dedicated collection of rack mounted computers
- As a job manager and resource scheduler for a collection of desk-top computers
- As a job manager and a resource scheduler for a collection of batch/grid/cloud systems
- As a job manager and resource scheduler for all of the above

Everything "looks" and is treated like a job

April 19, 2012, 9:02 a.m. EDT

Cycle Computing Ramps Global 50,000-Core Cluster for Schrodinger Molecular Research Utility Supercomputing Leader Facilitates Massive Cluster for Computational Drug Discovery

NEW YORK, NY, Apr 19, 2012 (MARKETWIRE via COMTEX) -- Cycle Computing provisioned a 50,000-core utility supercomputer in the Amazon Web Services (AWS) cloud for Schroedinger and Nimbus Discovery to accelerate lead identification via virtual screening. This milestone -- the largest of its kind -- is Cycle Computing's fifth massive cluster in less than two years on the heels of a 30,000 cluster in October 2011, illustrating Cycle's continued leadership in delivering full-featured and scalable cluster deployments. Cycle Computing revealed the cluster creation during today's opening keynote at the AWS Summit in New York City.

Are EC2 Spot instances a Grid, a Cloud or just a Distributed Processing System where resources come and go at (local) will?

The Open Science Grid (OSG)

Open Science Grid

THE UNIVERSITY of WISCONSIN MADISON

"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

THE UNIVERSITY of WISCONSIN MADISON

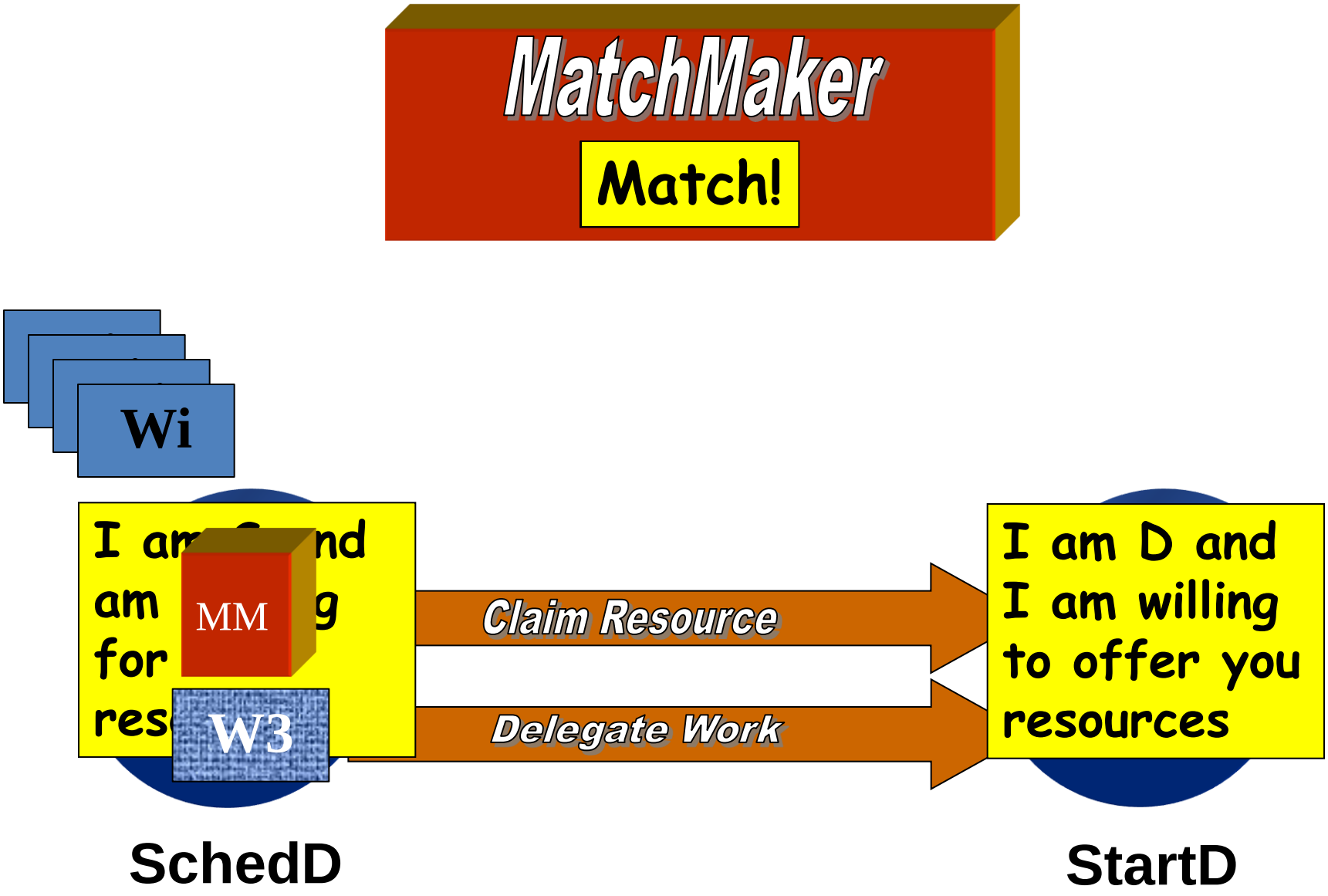
The OSG addresses these challenges by following a framework that is based on four underlying principles:

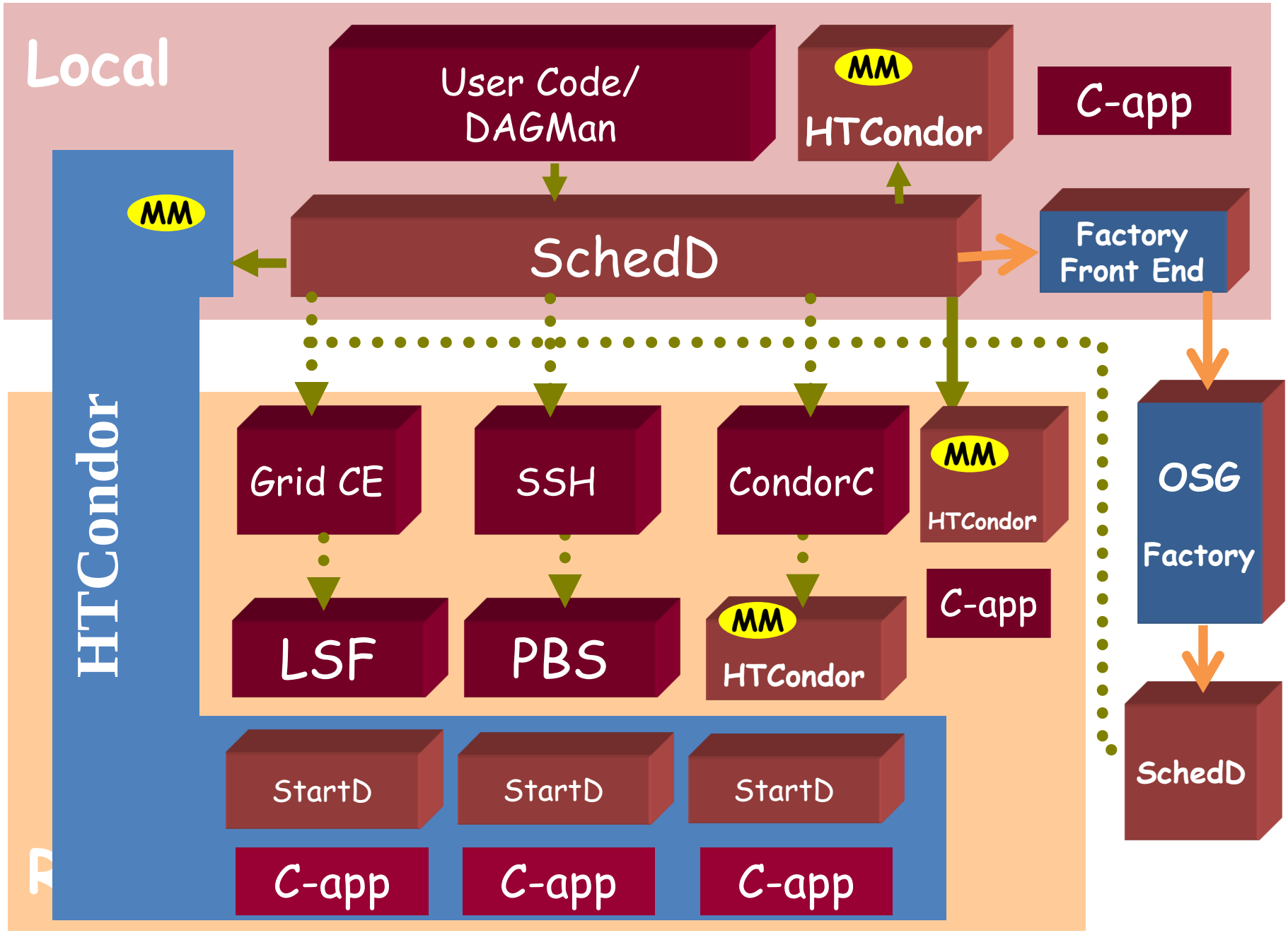
- *Resource Diversity*
- *Dependability*
- *Autonomy*
- *Mutual Trust*

"This dependability needs to be maintained while the services and their software implementations change to meet new needs and incorporate new technologies"

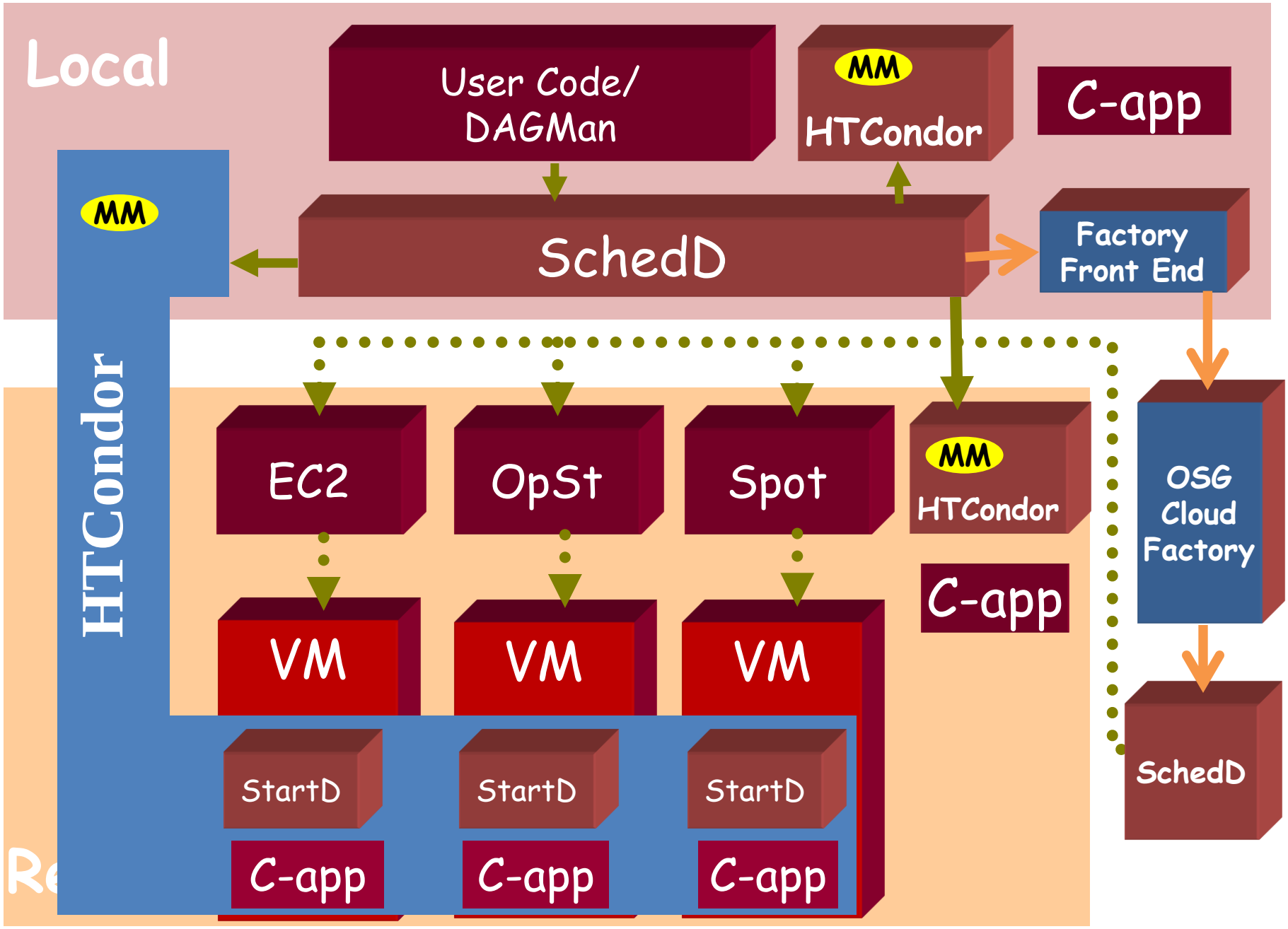
You may have ONE local submit machine managing 100K jobs on 10K remote machines

In HTCondor we use a two phase matchmaking process to first allocate a collection of resources to a requestor and then to select a task to be delegated for execution within the constraints of these resources


MatchMaker
Match!
Wi
I am
nd
am
g
for
res
MM
W3
Claim Resource
Delegate Work
I am D and
I am willing
to offer you
resources
SchedD
StartD


Local
User Code/
DAGMan
MM
HTCondor
C-app
MM
SchedD
Factory
Front End
HTCondor
Grid CE
SSH
CondorC
MM
HTCondor
OSG
Factory
C-app
LSF
PBS
MM
HTCondor
SchedD
StartD
StartD
StartD
C-app
C-app
C-app

The OSG GildeIn factory uses the SchedD as a resource provisioning agent on behalf of the (local) SchedD. It decides when, from where and for how long to keep an acquired resource.


Local
User Code/
DAGMan
MM
HTCondor
C-app
MM
SchedD
Factory
Front End
HTCondor
EC2
OpSt
Spot
MM
HTCondor
C-app
OSG
Cloud
Factory
VM
VM
VM
StartD
StartD
StartD
C-app
C-app
C-app
SchedD

The main challenge is to know in advance how much an application needs, to monitor how much an application actually consumes to know how much is available and to react accordingly

We use networks for control, we use networks to move executables and we use networks to move data. The application may use networks internally to its work.

When talking about networks, everyone always thinks of "bandwidth", but indeed there is much more to consider....

DEFINITIONS

"Intermediary" - Anyone between the client and the server. Could the operating systems on either side, the network cards on either side, routers, switches, NATs, and more.
"server" - The side listening for an inbound TCP connection or UDP packet. A given process might be both a server and a client.
"client" - The side initiating a TCP connection or sending a UDP packet. A given process might be both a server and a client.

Bandwidth - Can you transmit required data quickly enough to meet your needs? You might be constrained by the physical links as well as any intermediaries. For example, many firewalls process packets more slowly than their network connections otherwise support. Various layers of the system itself may limit bandwidth; security (encryption, decryption, checksums) in particular can easily be expensive. Security can also increase the amount of traffic necessary; authentication of both sides can easily add multiple messages to ultimately send a single small message.

CPU - Can your system make requests quickly enough? Can your system take advantage of multiple CPU cores in a single system to manage load? How much CPU does your security subsystem require; security (encryption, decryption, checksums) can use a lot.

Memory – May be limited by physical RAM and swap, kernel configuration, user-level limits, cgroup limits, or per-process limits. Adding a security subsystem will require more memory.

- Process memory - Each network connection requires some memory in your process. Your library providing the networking interface is almost certainly doing memory allocation on your behalf.
- Kernel memory - Each network connection, including pending, requires some memory from the kernel.

File descriptors - Each connection requires an FD, and listening for incoming connections is another FD. FDs are a finite resource on the client and server.

Multiple layers can impose limits: per process limits (ulimit), per user limits, per process group limits (by cgroups or similar), configurable system-wide limits (/proc/sys/fs/file-max on Linux), or technical limits (Only $2^{\wedge 32}$ FDs can be described in a single program on Linux. We believe Linux can only manage $2^{\wedge 32}$ total FDs).

Ports - Only 65535 are available on a given network interface. In practice, the available number will be much smaller: some will be in use by other processes, some are unavailable to non-root processes, and the system configuration will likely limit the range further (/proc/sys/net/ipv4/ip_local_port_range on Linux controls the ephemeral ports). Closed connections may continue to hold ports for a while to reduce the risk of port-reuse (TCP's TIME_WAIT state). In some cases ports can be shared, but this necessitates adding additional identifying information for each TCP connection or UDP packet.

All computers behind a NAT share a finite number of ports for all connections to hosts outside of the NAT.

Firewall state - A firewall/router/NAT has limited resources to manage connections that traverse it. It might have limits on simultaneous connections, simultaneous connections being initiated, bandwidth, RAM, or others. If the intermediary runs out of resources, a wide variety of undesirable things might happen: the intermediary may stop processing anything, blocking all traffic; it may break existing connections; it may reject new connections; it may stop processing firewall rules.

What does HTCondor offer today for network resoures

The SchedD can manage the allocation of data transfer connections to users/jobs while monitoring overall I/O and networking activity

The SharedPort daemon reduces the number of ports used by an HTCondor machine to one

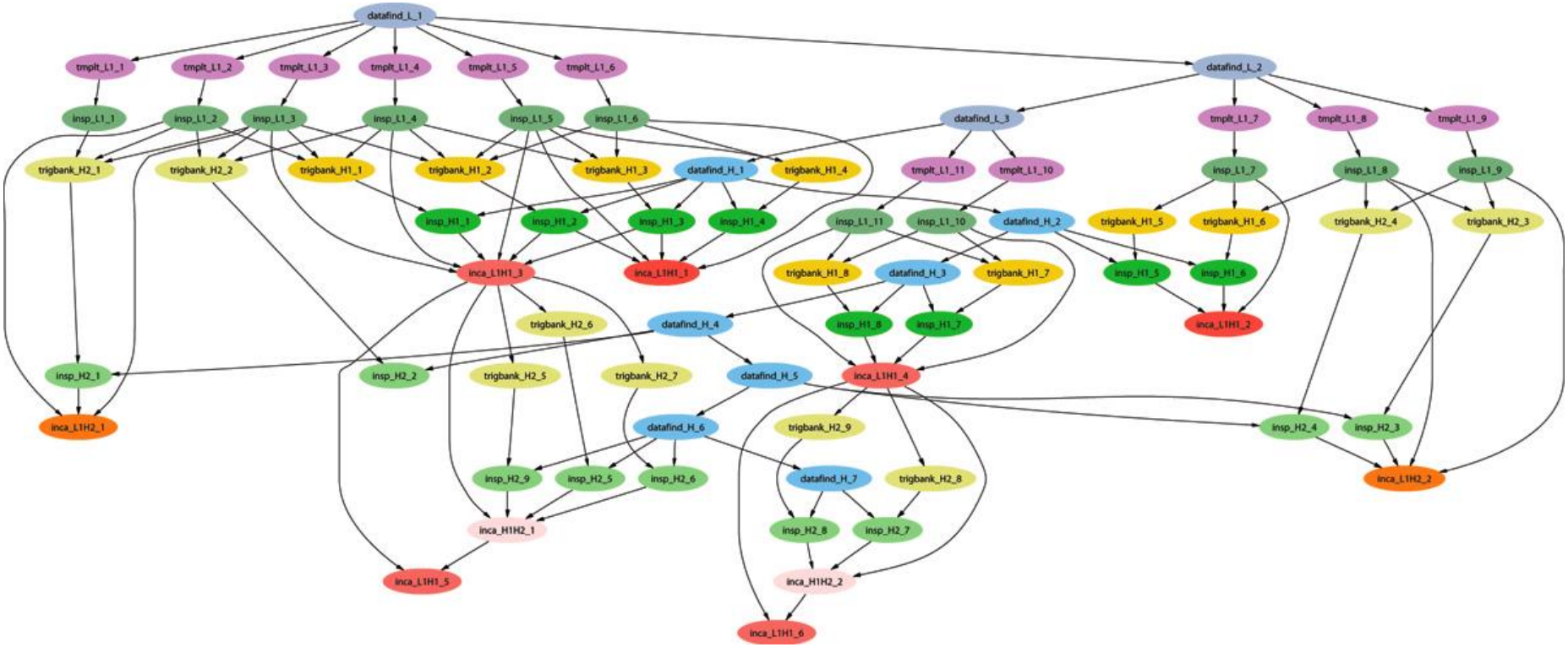
by reversing TCP connections the Condor Connection Broker (CCB) reduces the number of (outgoing) Ports used by a SchedD

HTCondor monitors FD consumption and collects HostName resolution statistics

Know what you need, what you use and what is available!

Using the power of Directed Acyclic Graphs (DAGs) to support declarative automation of interdependent tasks.

Example of a LIGO Inspiral DAG (Workflow)



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)**.

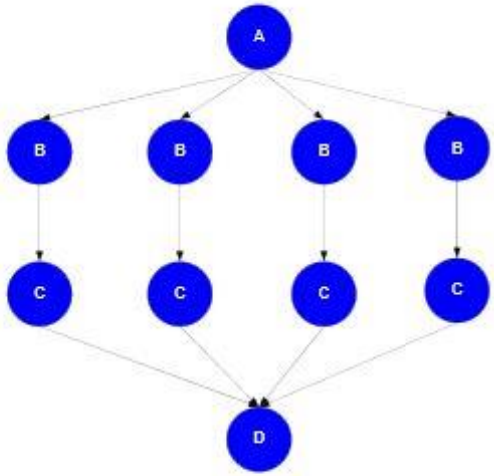
P.S. These are lower bounds as I did not perform an exhaustive survey/search, but they are probably close.

Thanks.

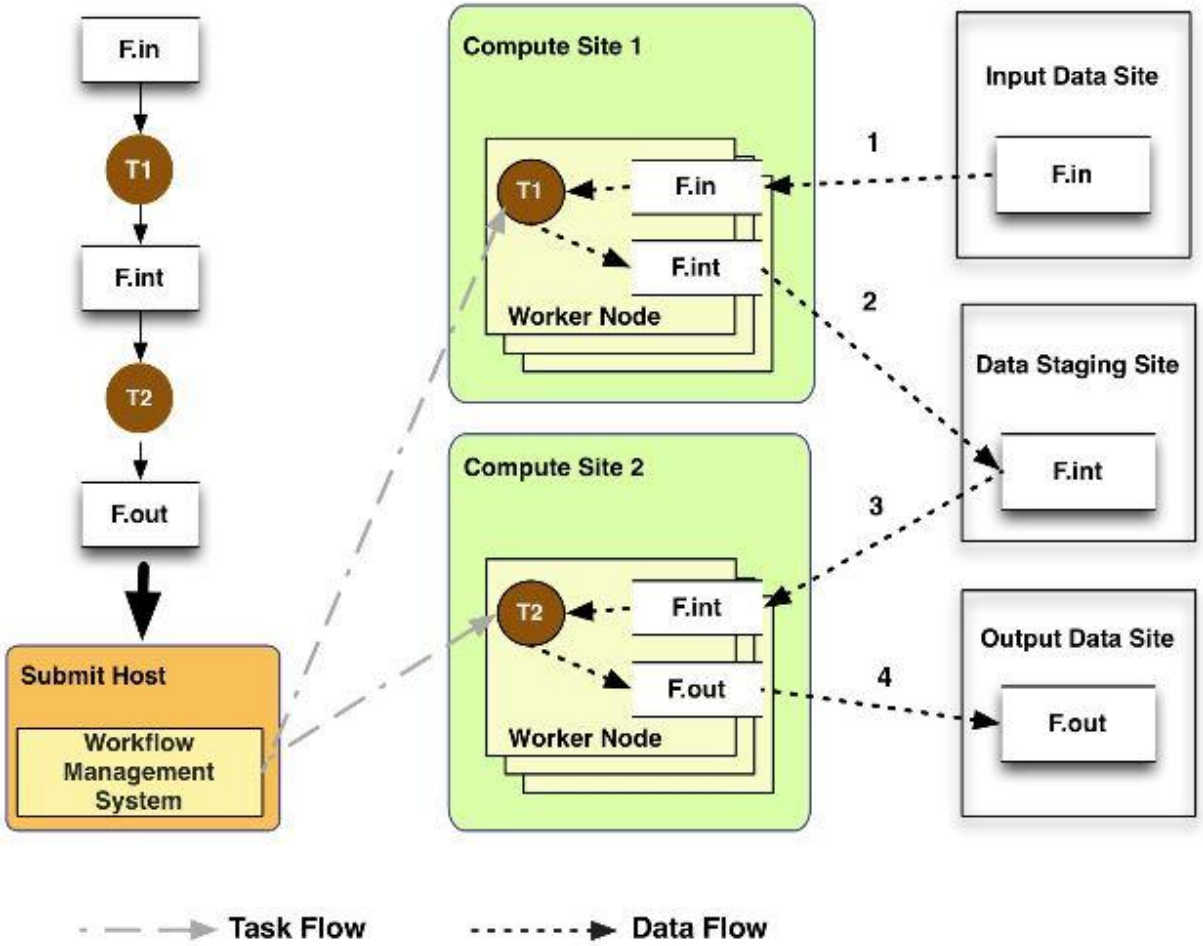
Using the power of the "data-flow" model support planning and enable automation

Pegasus Workflow Management System

- **NSF funded project since 2001**
  - Developed as a collaboration between USC Information Sciences Institute and the Condor Team at UW Madison
- **Builds on top of Condor DAGMan.**
- **Abstract Workflows - Pegasus input workflow description**
  - Workflow "high-level language"
  - Only identifies the computation, devoid of resource descriptions, devoid of data locations
  - File Aware
- **Pegasus is a workflow "compiler" (plan/map)**
  - Target is DAGMan DAGs and Condor submit files
  - Transforms the workflow for performance and reliability
  - Automatically locates physical locations for both workflow components and data
  - Collects runtime provenance



General Workflow Execution Model



- Most of the tasks in scientific workflow applications require POSIX file semantics
  - Each task in the workflow opens one or more input files
  - Read or write a portion of it and then close the file.

- Input Data Site, Compute Site and Output Data Sites can be co-located
  - Example: Input data is already present on the compute site.

Homework – Mechanisms to manage opportunistic storage.