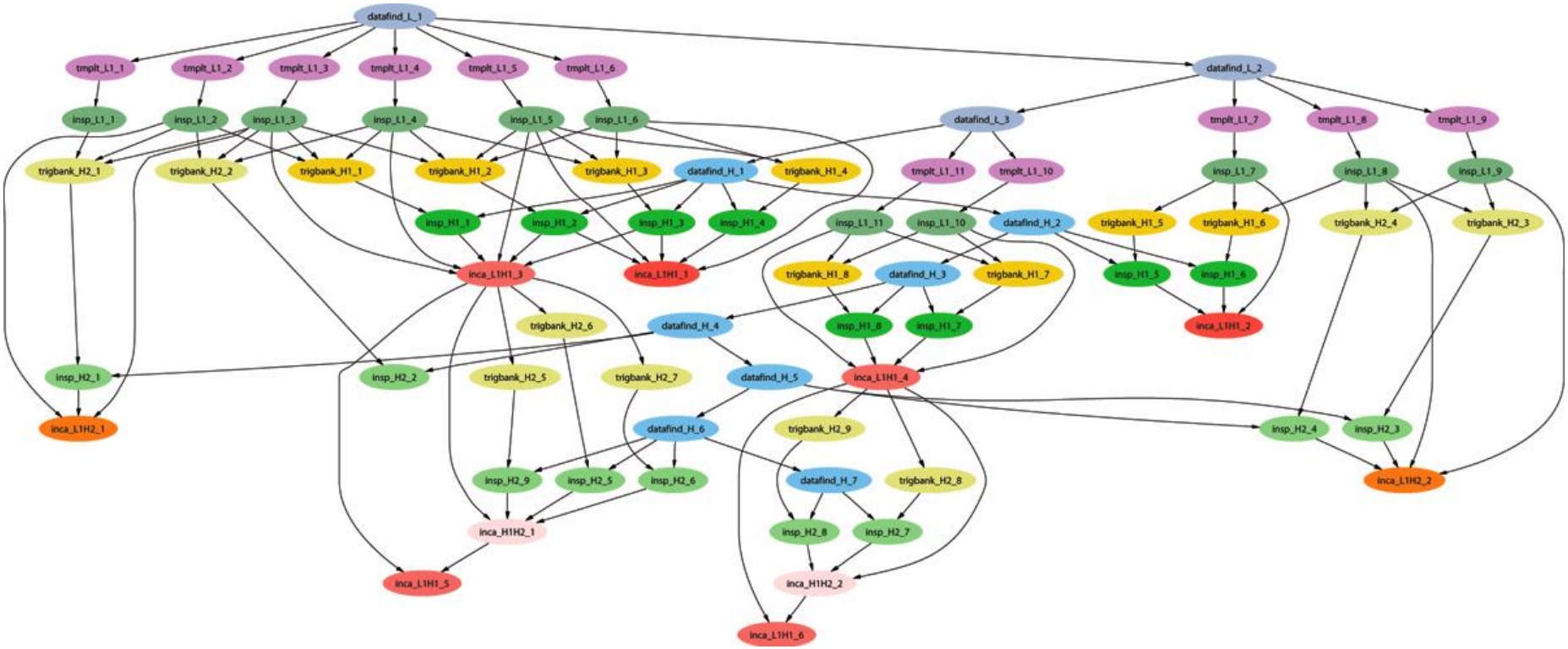
What defines a workload as High Throughput?

Miron Livny
John P. Morgridge Professor of Computer Science
Wisconsin Institutes for Discovery
University of Wisconsin-Madison

Researcher view of computing:

Place $y = F(x)$ at L!

Example of a LIGO Inspiral DAG



To do the same with `htcmap`, we just use the `map` function it provides instead. Note that `htcmap` has persistence for completed jobs, so if you get a `clusterid` of `None`, you already have the outputs for all of your inputs cached.

In [ ]:
```
result = htcmap.map(double, range(20))
result
```

That function returns a `MapResult` which we can use to get information about the running jobs.

For example, we can call its `tail` method to tail the cluster log. Note that this runs forever, so you'll need to interrupt the Jupyter kernel (black square along the top bar) to run the next cell.

In [ ]:
```
result.tail()
```

2 CARMI

The Condor Application Resource Management Interface (CARMI) provides services for writing parallel applications in an environment with dynamic resources. CARMI, therefore, allows an application to exploit new resources which become available at runtime, and aids an application in detecting and managing resource loss. These services can be used in a dedicated environment to utilize resources which are freed by other applications, or to allow a scheduling mechanism to revoke resources from a running application. In an opportunistic environment, CARMI permits an application to grow to new resources as they become available, and cope with resources being reclaimed by their owner.

J. Pruyne and M. Livny, ``Parallel Processing on Dynamic Resources with CARMI,'' in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Springer-Verlag, 1995.

A simple DAG for y=F(x)→L

1. Allocate (size(x)+size(y)+size(F)) at SE(i)
2. Move x from SE(j) to SE(i)
3. Place F on CE(k)
4. Compute F(x) at CE(k)
5. Move y to L
6. Release allocated space

Storage Element (SE); Compute Element (CE)

Data Placement

Management of storage space and bulk data transfers play a key role in the end-to-end performance of an application.

- Data Placement (DaP) operations must be treated as "first class" jobs and explicitly expressed in the job flow
- Fabric must provide services to manage storage space
- Data Placement schedulers are needed.
- Data Placement and computing must be coordinated
- Smooth transition of CPU-I/O interleaving across software layers
- Error handling and garbage collection

Miron Livny
John P. Morgridge Professor of Computer Science
Director Center for High Throughput Computing
University of Wisconsin Madison

It is all about Storage Management, stupid!

(Allocate **B** bytes for **T** time units)

- We do not have tools to manage storage allocations
- We do not have tools to schedule storage allocations
- We do not have protocols to request allocation of storage
- We do not have means to deal with “sorry no storage available now”
- We do not know how to manage, use and reclaim opportunistic storage
- We do not know how to budget for storage space
- We do not know how to schedule access to storage devices

Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017-2020

AUTHORS

Committee on Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science in 2017-2020; Computer Science and Telecommunications Board; Division on Engineering and Physical Sciences; National Academies of Sciences, Engineering, and Medicine

"… many fields today rely on **high-throughput computing** for **discovery**."

"Many fields **increasingly** rely **on high-throughput computing**"

"The members of **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."

Claims for "benefits" provided by Distributed Processing Systems

- 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

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

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

Focus on the
problems that are
unique to HTC
not the latest/greatest
technology

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

Capability
vs.
Capacity

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**.

Obstacles to HTC

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

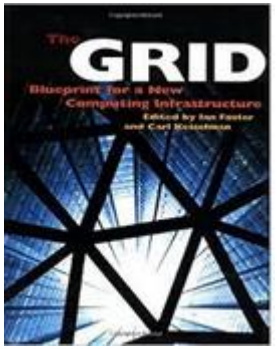
High Throughput Computing requires automation as it is a 24-7-365 activity that involves large numbers of jobs

FLOPY ≠ *(60*60*24*7*52)*FLOPS*

100K Hours*1 Job ≠ 1 H*100K J

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*".**

What can Grid computing do for you?

My Application …

Study the behavior of **F**(*x*,*y*,*z*) for 20 values of *x*, 10 values of *y* and 3 values of *z* (20*10*3 = 600)

- **F** takes on the average 3 hours to compute on a "typical" workstation (total = 1800 hours)
- **F** requires a "moderate" (128MB) amount of memory
- **F** performs "little" I/O - (*x*,*y*,*z*) is 15 MB and **F**(*x*,*y*,*z*) is 40 MB

Step I - get organized!

- Write a script that creates 600 input files for each of the (x,y,z) combinations
- Write a script that collects the data from the 600 output files
- Turn your workstation into a "Personal Condor"
- Submit a cluster of 600 jobs to your personal Condor
- Go on a long vacation … (2.5 months)

A Condor Job-Parallel Submit File

```
executable   = worker
requirement = ((OS == "Linux2.2") && (Memory >= 64))
rank         = KFLOP
initialdir   = worker_dir.$(process)
   input          = in
   output         = out
   error          = err
   log            = Condor_log
queue 600
```

HTC is about many **jobs**, many **users**, many **servers**, many **sites** and (potentially) long running **workflows**

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

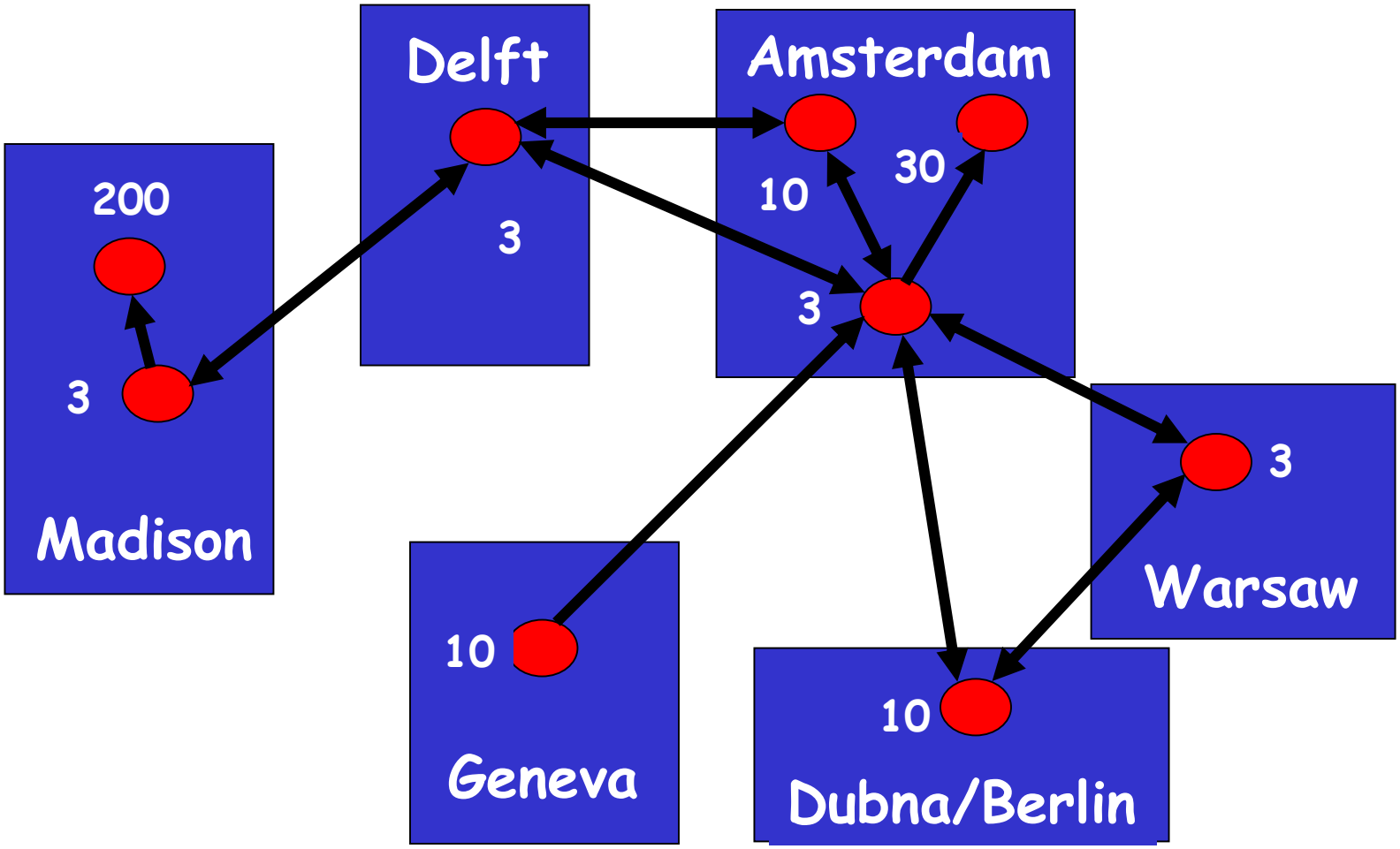
Use wide-area networks to transfer batch jobs between Condor systems

- 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

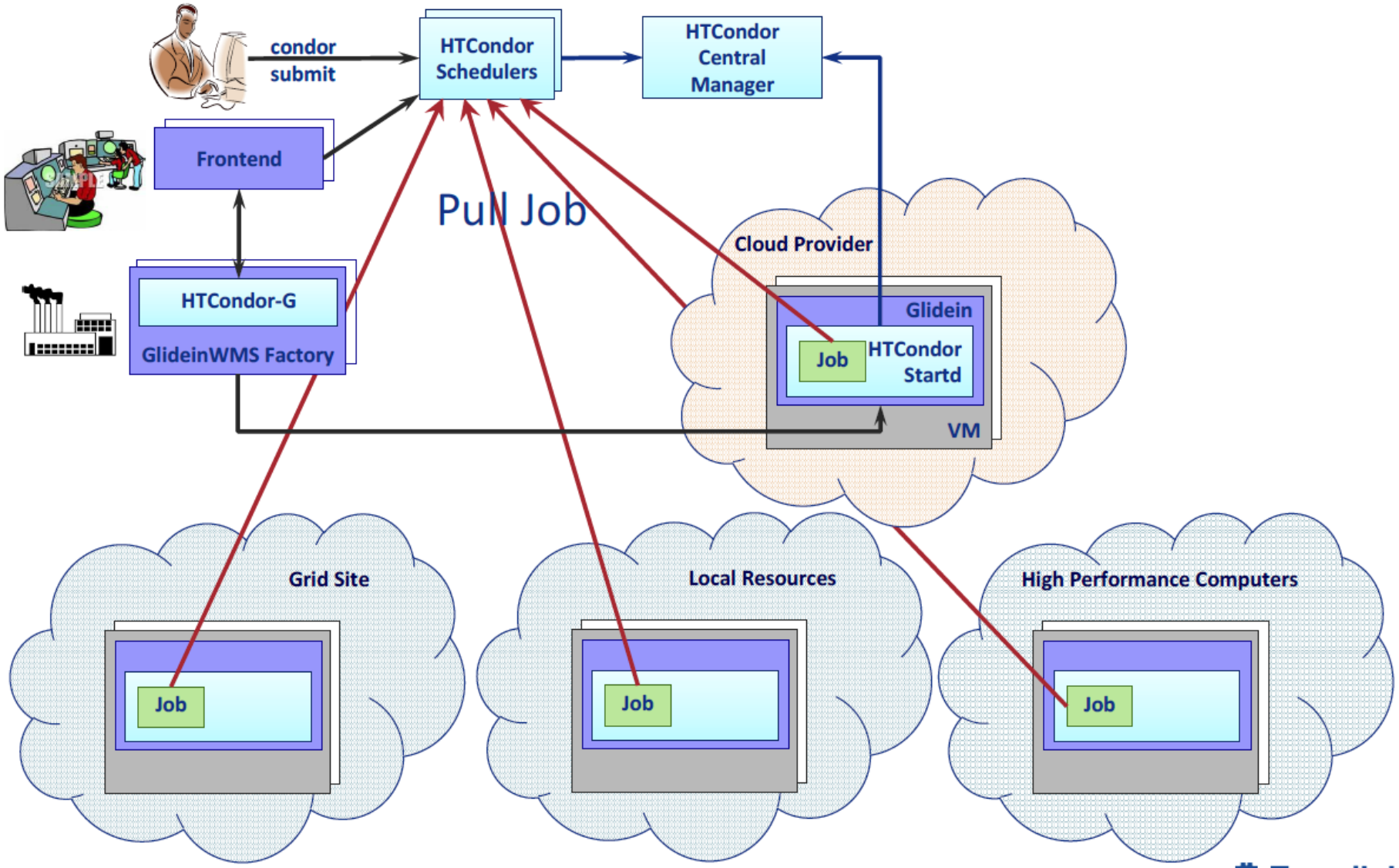
1994 Worldwide Flock of Condors



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

Submit locally (queue and manage your jobs/tasks locally; leverage your local resources) **and run globally** (acquire any resource that is capable and willing to run your job/task)

HEPCloud – glideinWMS and HTCondor



Fermilab