The Principles of HTC

Miron Livny
Wisconsin Institutes for Discovery
University of Wisconsin-Madison
How a single neutrino just helped crack a 100-year-old cosmic ray mystery

An enormous ice cube at the South Pole is revealing mysteries about the universe.

By Brian Resnick | @b_resnick | brian@vox.com | Jul 12, 2018, 11:00am EDT

In an artist's depiction that is very, very not-to-scale, a blazar is shown shooting a beam of cosmic rays at the Earth. | IceCube/NASA
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
Focus on the needs and expectations of researchers
HTCondor at DreamWorks

Collin Mehring
Using HTCondor Since 2011
How do we have HTCondor configured?

- All DAG jobs
  - Many steps involved in rendering a frame
- GroupId.NodeId.JobId instead of ClusterId
  - Easier communication between departments
- No preemption (yet)
  - Deadlines are important - No lost work
  - Checkpointing coming soon in new renderer
- Heavy use of group accounting
  - Render Units (RU), the scaled core-hour
  - Productions pay for their share of the farm
- Execution host configuration profiles
  - e.g. Desktops only run jobs at night
  - Easy deployment and profile switching
- Load data from JobLog/Spool files into Postgres, Influx, and analytics databases

Quick Facts

- Central Manager and backup (HA)
  - On separate physical servers
- One Schedd per show, scaling up to ten
  - Split across two physical servers
- About 1400 execution hosts
  - ~45k server cores, ~15k desktop cores
  - Almost all partitionable slots
- Complete an average of 160k jobs daily
- An average frame takes 1200 core hours over its lifecycle
- Trolls took ~60 million core-hours
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 Chapter 1 verse 9

We are driven by Principals (¬ Hype)

Source: Gartner (August 2014)
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.
The paradigm shift of 70’s – computing hardware packaged and sold in small units
The paradigm shift of 00’s – computing capacity leased by the minute when needed
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
DISTRIBUTED COMPUTING BASICS FOR BIG DATA

By Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman

If your company is considering a big data project, it’s important that you understand some distributed computing basics first. There isn’t a single distributed computing model because computing resources can be distributed in many ways.
Definitional Criteria for a Distributed Processing System


– 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

Minimize wait (job/task queued) while Idle (a resource that is capable and willing to serve the job/task is running a lower priority job/task)
When each resource has its own queue, when should I stay at the current queue and wait and when 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’. ... “

In 1985 we extended the scope of the distributed load balancing problem to include “ownership” of resources.
Should I share my resource and if I do with whom and when?
Mine
Yours
Yours
Yours
Now you have a community of customers who are motivated to share and act as consumers, providers or both.
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
by Alan Beck, editor in chief

06.27.97

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

“Many fields increasingly rely on high-throughput computing”
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
Obstacles to HTC

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

1997
The Open Science Grid (OSG) national fabric of distribute HTC services
“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.”
1.60B core hours in 12 months!

Almost all jobs executed by the OSG leverage HTCondor technologies:

- Condor-G
- HTCondor-CE
- Basco
- Condor Collectors
- HTCondor overlays
- HTCondor pools
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)
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}
• **Job owner identity is local**
  • Owner identity should never “travel” with the job to execution site
  • Owner attributes are local

• **Name spaces are local**
  • File names are locally defined

• **Resource acquisition is local**
  • Submission site (local) is responsible for the acquisition of all resources
• “external” forces moved us away from this “pure” local centric view of the distributed computing environment.

• With the help of capabilities (short lived tokens) and reassignment of responsibilities we are committed to regain full local control.

• Handing users with money (real or funny) to acquire commuting resources helps us move (push) in this positive direction.
Using Directed Acyclic Graphs (DAGs) to support declarative automation of interdependent tasks
2017 Nobel Prize in Physics

High Throughput Computing helps LIGO confirm Einstein’s last unproven theory

March 8, 2016 | By Brian Mattmiller | For news media

Two Black Holes Merge Into One

A computer simulation shows the collision of two black holes, a tremendously powerful event detected for the first time ever by the Laser Interferometer Gravitational-Wave Observatory, or LIGO. Source: Simulating eXtreme Spacetimes (SXS)
When a workflow might consist of 600,000 jobs, we don’t want to rerun them if we make a mistake. So we use DAGMan (Directed Acyclic Graph Manager, a meta-scheduler for HTCondor) and Pegasus workflow manager to optimize changes,” added Couvares. “The combination of Pegasus, Condor, and OSG work great together.” Keeping track of what has run and how the workflow progresses, Pegasus translates the abstract layer of what needs to be done into actual jobs for Condor, which then puts them out on OSG.
Example of a LIGO Inspiral DAG (Workflow)
HTC is about sharing across many jobs, many users, many servers, many sites and (potentially) long running workflows
A job submitted to a batch service consists of an Acquisition Request (AquR) and a Job Description (JobD).

The Provision Manager (Pman) of the service provisions the resources and then runs the job on theses resources via the Job Launcher (JaL).
Most batch services manage a static collection of resources
HTCondor uses a matchmaking process to dynamically acquire resources.

HTCondor uses a matchmaking process to provision them to queued jobs.

HTCondor launches jobs via a task delegation protocol.
**Figure 1:** The HTC Framework: (1) A user submits job, composed of an AcqR and a JobD. (2) The AMan requests a resource compatible with an AcqR. (3) The MarketPlace offers a compatible Resource to the batch service. (4) The Pman selects a job description and Resource to send to (5) the JLa, which runs the job.
Submit Locally Run Globally

OSG integrates computing across different resource types and business models.
Traditional (low frequency) Capacity Planning
Turning $s$ into computing power

- Collect workload characteristics and customer (performance) metrics
- Understand the cost-performance profile of the hardware and software options
- **Acquire** (select, purchase, install) the resources and place them under the control of a batch service
- Live with your decision for (5-8) years
Next generation (High frequency) Capacity Planning when resources can be rented by the minute or by the hour
Researcher or VOs may have ...

- **Resources** they own and therefore fully control
- An **allocation** of resources on shared campus/national computing facility
- "**Fair Share**" privileges on shared campus/national computing facilities
- **Opportunistic Resources** provided by collaborators
- **Funding** to purchase resources from a commercial cloud provider
Commercial clouds offer to individuals with money ...

• **Unbounded** on demand capacity for (almost) as long as needed

• A **variety** of cost/performance option for processing and storage resources

• **Dynamic cost** structures that track demand and supply

• **Diverse** (and competing) suppliers of computing resources and associated services
**SC16 Demo: On Demand Doubling of CMS Computing Capacity**

Joint project

**HEPCloud** (Fermilab), **HTCondor** (UW-Madison),

- **HEPCloud** provisions **Google Cloud** with **HTCondor** in two ways
  - **HTCondor** talks to Google API
  - Resources are joined into **HEP HTCondor** pool

- Demonstrated sustained large scale elasticity (>150K cores) in response to demand and external constraints
  - Ramp-up/down with opening/closing of exhibition floor
  - Tear-down when no jobs are waiting

730,172 jobs consumed 6.35M core hours to produce 205M simulated events (81.8 TB)

Total cost ~$100K
500 TB were placed in Google Cloud in advance. 80TB where moved back to Fermi.

- $8.6k network egress
- $8.5k disk attached to VMs
- $3.5k cloud storage for input data
HEPCloud is an R&D project led by the Fermi computing division
HEPCloud – glideinWMS and HTCondor

![Diagram of HEPCloud system showing HTCondor Schedulers, Frontend, HTCondor-G, GlideinWMS Factory, HTCondor Central Manager, Cloud Provider, Glidein, and Job execution on Grid Site, Local Resources, and High Performance Computers.](image-url)

- **Frontend** submits jobs to **HTCondor Schedulers**.
- **HTCondor-G** submits jobs to **GlideinWMS Factory**.
- **Glidein** submits jobs to **Cloud Provider**.
- Jobs are executed on **Grid Site**, **Local Resources**, and **High Performance Computers**.
HEPCloud Architecture

1. A workflow is submitted to the Facility Interfaces
2. Workflow is authenticated and authorized
3. Decision Engine queries for job manifests to determine job requirements and characteristics.
4. Decision Engine queries for API level authorization
5. Decision Engine queries for available resources
6. If required, a request for resources is made to the Provisioner.
7. Provisioner requests resources
8. Provisioned Resources join Facility Pool

Monitoring

Legend:
- A: Authentication and Authorization I/O
- M: Monitoring I/O
Here is what the OSG offers today with the support of HTCondor technologies
I am D and I am willing to offer you resources.

I am S and I am looking for resources.

Match!

SchedD

StartD

Claim Resource

Delegate Work

Wi

W3

MM
HTCondor 101

• Jobs are submitted to the HTCondor SchedD
• A job can be a Container or a VM
• The SchedD can Flock to additional Matchmakers
• The SchedD can delegate a job for execution to a HTCondor StartD
• The SchedD can delegate a job for execution to another Batch system.
• The SchedD can delegate a job for execution to a Grid Compute Element (CE)
• The SchedD can delegate a job for execution to a Commercial Cloud
Welcome to the HTC Community
Timeline of projects

- MONARC project CERN
- LHC Approval
- Hoffmann Review
- Computing TDRs
- LHC Run 1
- Run 2
- WLCG

- Enslow’s DPS paper
- Condor Deployed
- My PhD
- "Trillium" = PPDG+GriPhyN+IVD GL

Timeline:
- 1978
- 1983
- 1985
- 1992
- 1993
- 1994
- 1995
- 1996
- 1997
- 1998
- 1999
- 2000
- 2001
- 2002
- 2003
- 2004
- 2005
- 2006
- 2007
- 2008
- 2009
- 2010
- 2011
- 2012
- 2013
- 2014
- 2015

25 July 2014

Ian Bird; Varenna