High Throughput Computing
with
HTCondor

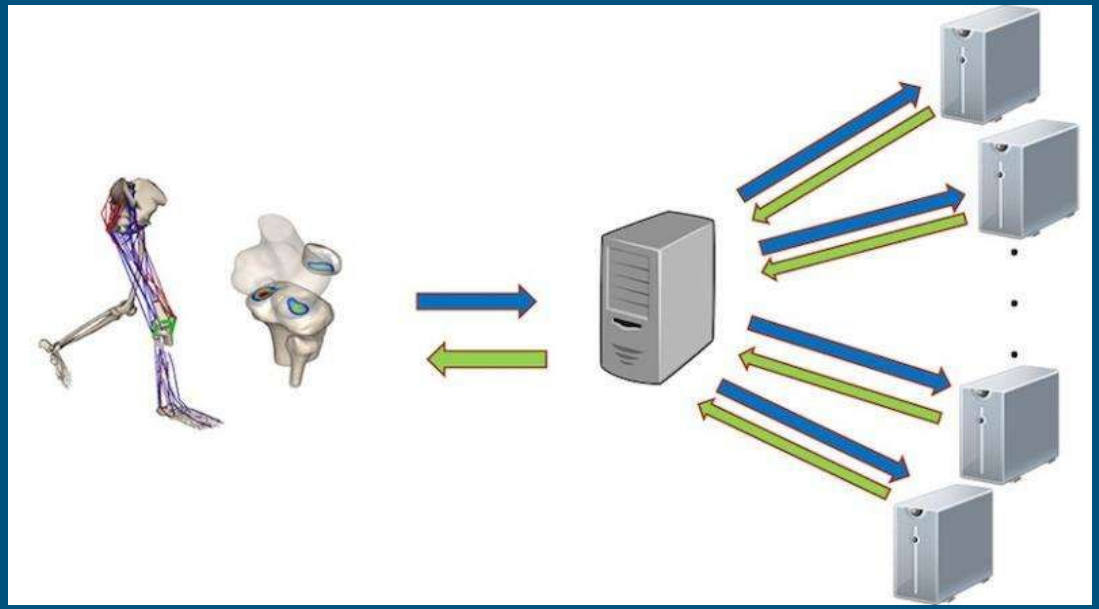
Presented by:
Sabbir Ahmed Sibli (20266027)
Nuray Jannat (20266028)
Hasibul Islam Sifat (20266008)
Taniya Sultana Jabin (20266016)
Sahiba Tasneem (20266022)

Table of Contents

Intro: High Throughput Computing (HTC)

- HPC deals with one particular large problem.
- HTC deals with problem which can be broken into many smaller independent problems.
- Specifically, HTC means getting lots of work done in a specific time unit.
- HTC distributes tasks over many computers (e.g., idle desktop computers, dedicated servers, or cloud-based resources.

Intro: High Throughput Computing (HTC) contd.



High-throughput computing resources speed up the process of knee modeling simulations by sending jobs to many computers rather than just one. (Credit: University of Wisconsin-Madison)

Intro: HTCondor

- HTCondor "has enabled ordinary users to do extraordinary computing"
- It is a Distributed-batch computing system
- Users submit a job to HTCondor, then HTCondor chooses when and where to run the job based upon the job requirement and available worker machines.
- It monitors the jobs' progress, and notifies the user upon completion.
- The development of HTCondor started in 1998 by University of Wisconsin-Madison.

Intro: HTCondor contd.

HTCondor – an open-source workload management software for HTC jobs (developed at the University of Wisconsin – Madison)

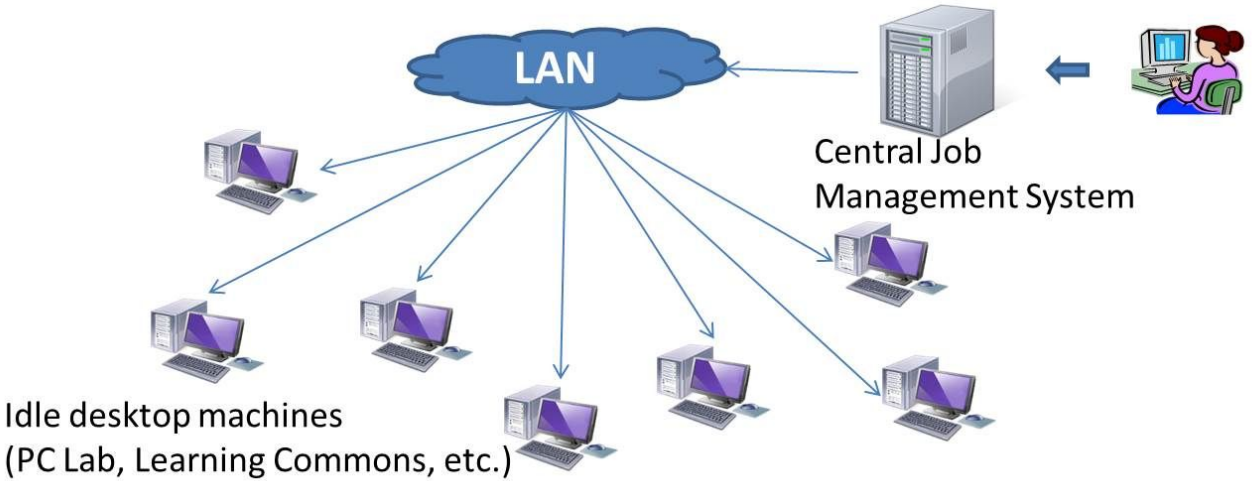


Image Credit: fakecineaste blog

HTCondor Terminologies

- Job
  - The Large task a user wants HTCondor to run.
- Batch/Shell Scripts
  - A file with instructions for a computer to execute.
- Submit Machine
  - One or more computers used by HTCondor to send jobs out at the start of a run and collect jobs at the end of a run.
- Class Ads
  - A language used by HTCondor to advertise resources on machines running HTCondor.
- Submit File
  - A file submitted by the user to the central manager. This file tells HTCondor the job requirements

Steps of Applying HTCondor

- Define the computing problem.
  - Choosing appropriate problems which suit to run with HTC rather than HPC.
- Discretize the problem into smaller jobs that may be run on HTCondor.
  - Breaking the large single task into many independent task that they can run independently.
- Process the Inputs
  - After discretizing, having a way to identify files by some naming convention.
- Run the job on HTCondor
  - Placement of a submit file and corresponding files on the central manager.
- Post process the Data.
  - This step consists of combining the output from the previously run HTCondor jobs. Often, this step is most easily done using a scripting language such as Python or R.

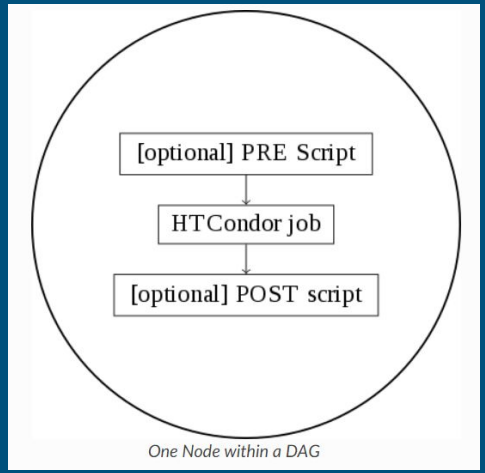
Managing Workflow: DAGMan Tool

- It is a HTCondor tool that allows multiple jobs to be organized in **workflows.**
- It is represented as a directed acyclic graph (DAG).
- A DAGMan workflow automatically submits jobs in a particular order, such that certain jobs need to complete before others start running.
- This allows the outputs of some jobs to be used as inputs for others, and makes it easy to replicate a workflow multiple times in the future.

Managing Workflow: DAGMan Tool contd.

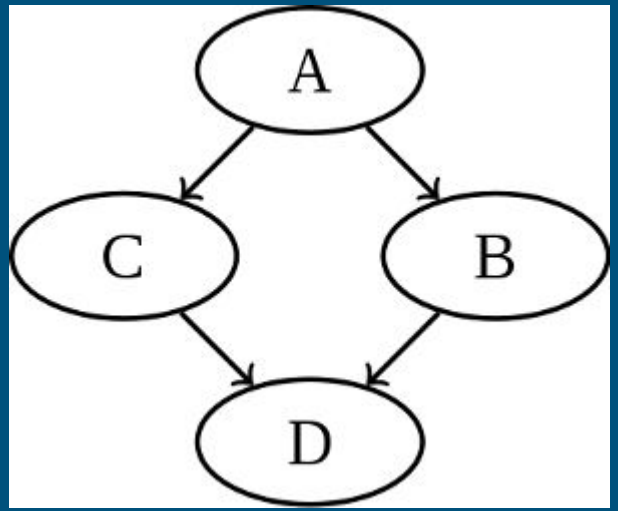
- A DAGMan workflow is described in a **DAG input file**.
- The input file specifies the nodes of the DAG. A **node** within a DAG represents a unit of work. It contains the following:

  - **Job:** An HTCondor job, defined in a submit file.
  - **Pre-Script:** A script that runs before the job starts. Typically used to verify that all inputs are valid.
  - **Post-Script:** A script that runs after the job finishes. Typically used to verify outputs and clean up temporary files.



One Node within a DAG

Managing Workflow: DAGMan Tool contd.



A simple diamond-shaped DAG. **Edge** in DAGMan describes a dependency between two nodes (i.e. parent, child).

A very simple DAG input file for this diamond-shaped DAG is:

```
# File name: diamond.dag

JOB  A  A.condor
JOB  B  B.condor
JOB  C  C.condor
JOB  D  D.condor
PARENT A CHILD B C
PARENT B C CHILD D
```

Exceptional Features of HTCondor

- Scalability
  - An HTCondor pool is horizontally scalable to hundreds of thousands of execute cores and a similar number of jobs.
- Security
  - HTCondor can be configured to use strong authentication and encryption between the services on remote machines used to manage jobs.
- No Special Program required to run
  - No special programming is required to use HTCondor.
- Flexible Policy Mechanisms
  - HTCondor allows users to specify very flexible policies for how they want jobs to be run.

“The purpose of computing is insight, not numbers.”

- Richard Hamming

References

1) Erickson, R.A., Fienen, M.N., McCalla, S.G., Weiser, E.L., Bower, M.L., Knudson, J.M. and Thain, G., 2018. Wrangling distributed computing for high-throughput environmental science: An introduction to HTCondor. PLoS computational biology, 14(10), p.e1006468.
2) https://htcondor.readthedocs.io/
3) Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience. 2005; 17:323–356.
4) http://fakecineaste.blogspot.com/2019/06/high-throughput-computing-htc.html

Thanks!

Image Source: Google