


# A Comparative Study of High Performance Computing on the Cloud



Lots of authors, including Xin Yuan  
Presentation by: Carlos Sanchez

# What is “The Cloud”?

---

- ▶ The cloud is just a bunch of computers connected over a network (usually Ethernet with internet protocols) which provides a service.
- ▶ Cloud networks usually have a layer of virtualization. When you connect to a cloud network, you’re connecting to a virtual machine, which could have been dynamically allocated on any of the cloud’s machines.
- ▶ Virtualization allows cloud networks to be more flexible in terms of location, size, etc. because the virtual hardware is abstracted from the physical hardware. The physical hardware can be changed/upgraded on the fly, but the virtual hardware (which users “connect” to) stays the same.



# High Performance Computing Clusters

---

- ▶ Computational problems such as simulations which require a great amount of computation time and which are highly parallelizable are usually run on High Performance Computing (HPC) clusters.
- ▶ HPC clusters are dedicated networks of computers with specialized hardware whose main purpose is to compute as much as possible in the shortest amount of time.
- ▶ These clusters are generally used for research and paid for with research funding, so users can access these resources for free



# HPC Clusters VS. Cloud Computing

---

- ▶ HPC clusters are focused on maximum performance for a very specific service (complex computations)
- ▶ Cloud networks are focused on maximum utilization of resources for a wide variety of services (and profit)
- ▶ HPC clusters for research are generally free thanks to funding, but your jobs are placed in a queue to be scheduled at the next available time
- ▶ Cloud networks provide a service for a fee, but that service is available “on-demand”



# Some Issues Addressed in the Paper

---

- ▶ HPC users believe that dedicated clusters are better than cloud platforms due to the communication overhead.
- ▶ Authors believe the difference is smaller than perceived and that total turnaround time and cost are better determining factors than a program's start and finish times.
- ▶ As research HPC clusters are free, a pricing model has been created in order to compare the prices of the on-demand cloud VS. the HPC cluster.
- ▶ Performance between the cloud and HPC clusters are also compared



# Which Cloud?

---

- ▶ Amazon's Elastic Compute Cloud (EC2) is used, as it is a highly successful and popular platform.
- ▶ However, in terms of high performance computing, successes are mixed.
- ▶ Most successful high performance computing applications that were run on EC2 were “embarrassingly parallel” programs
  - ▶ Embarrassingly parallel programs are programs which require little or no effort to parallelize.



# Why The Cloud “Doesn’t Work” For HPC

---

- ▶ Because cloud platforms use Ethernet for inter-connectivity, latency and bandwidth are an issue
  - ▶ Dedicated HPC clusters use Infiniband
- ▶ Virtualization causes computation overhead, and the relative location of virtual machines could be an issue
  - ▶ If the machines are virtual and can be anywhere, their physical proximity isn’t necessarily close or within a small number of hops, which could cause even more communication issues.



# Why The Cloud *Could* Work

---

- ▶ It isn't fair to compare a paid on-demand service which is available 24/7 to the traditional HPC cluster based only on execution time.
- ▶ Traditional HPC clusters can have a significant queue wait time, especially during peak usage and for jobs which require many nodes. The cloud is available on-demand.
- ▶ When comparing HPC clusters with the cloud, looking at total turnaround time and projected cost (should traditional HPC clusters stop being free) may show that the cloud is the way to go for tasks with certain computational, cost, and time requirements.





# What The Paper Gives Us

---

- ▶ Evaluate Amazon's EC2 and 5 HPC clusters from the Lawrence Livermore National Laboratory (LLNL) along the traditional axis of execution time using over 1000 cores.
- ▶ Develop a pricing model to evaluate (free) HPC clusters in node-hour prices based on system performance, resource availability, and user bias.
- ▶ Evaluate EC2 and HPC clusters along the axis of total turnaround time and total cost.



# EC2 Basics

---

- ▶ There are several EC2 instances with different computation and network capabilities. The paper chose the highest-end instance: “cluster compute eight extra large”, as it is intended for HPC applications
- ▶ There are also several ways of purchasing time. The paper chose “on-demand” pricing, where the purchaser receives access to the machine immediately.
- ▶ Default EC2 does not guarantee node proximity, but you can request it through a placement group. This was not used for the experiment.
  - ▶ HPC systems that use batch scheduling can’t guarantee physical proximity either.



# Quick Physical Specs

---

- ▶ Note: all clusters other than EC2 are machines at LLNL
- ▶ uDawn is a BlueGene/P system, and is one of the slowest machines.
- ▶ Cab and Sierra are newer machines, and are usually the highest performers.

Cluster	CPU speed (GHz)	Cache size (MB)	Memory size (GB)	Cores/Node	Interconnect Technology	Cost (\$/Hour)
Sierra	2.8	12	24	12	Infiniband QDR	—
Hera	2.3	0.5	32	16	Infiniband DDR	—
Cab	2.6	20	32	16	Infiniband QDR	—
Hyperion	2.4	6.0	12	8	Infiniband DDR	—
uDawn	0.85	0.02	2	4	3D Torus	—
EC2	2.59	20	23	16	10 GigE	2.4



# Program Setup

---

- ▶ Core 0 is avoided due to EC2 interrupt system
- ▶ Only half of the cores (or closest power of two) on a particular machine are used to reduce the overhead of uneven communication.
- ▶ No hyperthreading on Sandy-Bridge processors
- ▶ Strong scaling (all benchmark sizes are identical across different task counts)
- ▶ Wide variety of benchmarks; some are communication intensive, others are computation intensive.



# Tested Network Latency

---

- ▶ Cab and Sierra have the best stats thanks to Infiniband QDR
- ▶ EC2 has comparable throughput with uDawn, but the latency is far worse.
- ▶ Couldn't characterize the virtualization overhead, since there was no access to physical hardware which was identical to the virtual machines.
- ▶ Values for EC2 are consistent with other reported results

	Cab	Hera	Sierra	Hyperion	uDawn	EC2
Latency	1.61 $\mu$ s	2.23 $\mu$ s	1.58 $\mu$ s	1.91 $\mu$ s	2.92 $\mu$ s	55.15 $\mu$ s
Bandwidth	22.6 Gb/s	9.3 Gb/s	23.1 Gb/s	16.9 Gb/s	3.1 Gb/s	3.7 Gb/s

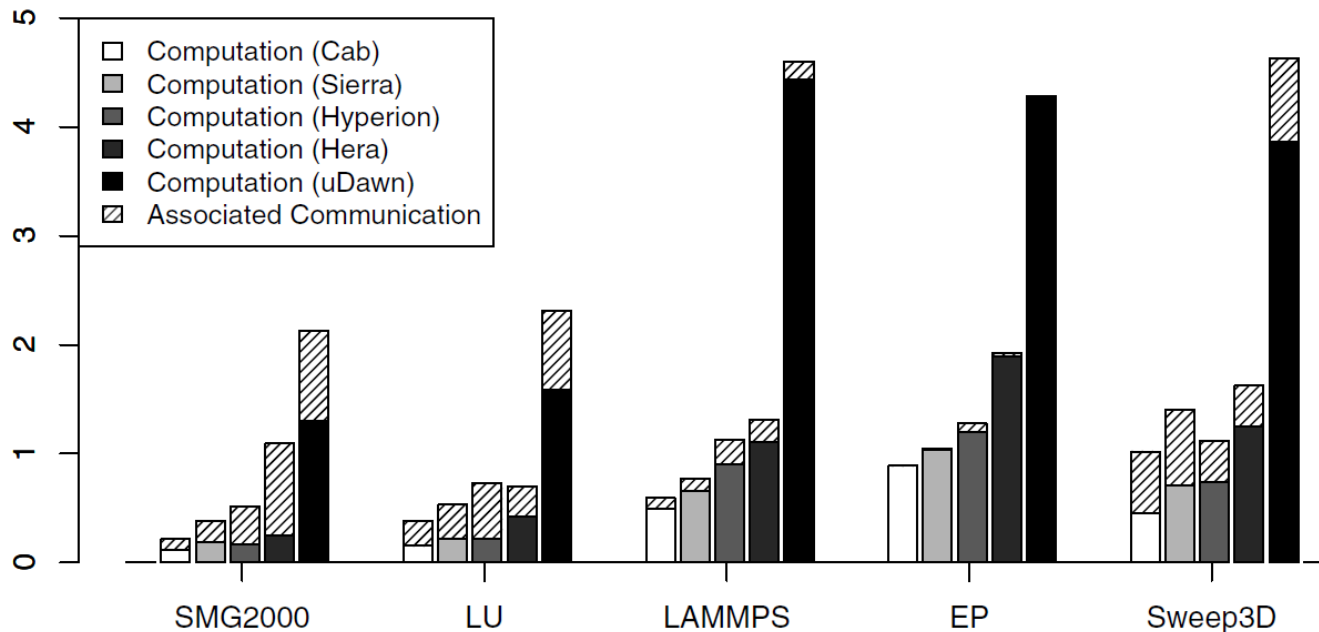


# Tested Overall Performance

- ▶ Normalized to uDawn (1 MPI task, computation only)

uDawn	Cab	Sierra	Hyperion	Hera	EC2
1.00	9.55	7.30	7.22	3.62	8.22

- ▶ Normalized to EC2 (1024 MPI tasks, execution time only)



# Overall Performance Cont.

---

- ▶ Results show that communication intensive benchmarks such as SMG2000 run quite a bit slower on EC2 than other machines (except uDawn).
- ▶ However, benchmarks such Sweep3D run better on EC2 than any other system.
- ▶ Not only that, but benchmarks such as LAMMPS have some machines performing better and some performing worse than EC2.
- ▶ On the grounds of only program execution time, the choice of whether to use the cloud or not is dependent on the qualities of the applications.



# Queue Wait Times

---

- ▶ Average queue wait times must be known in order to measure turnaround time.
- ▶ However, wait times vary depending on the cluster, job size, and maximum run time
- ▶ Estimating queue wait times isn't easy
  - ▶ The scheduling implementation isn't known
  - ▶ The wait time isn't necessarily linear with respect to the number of nodes or time requested.
  - ▶ Cluster utilization fluctuates depending on current jobs





## Queue Wait Times Cont.

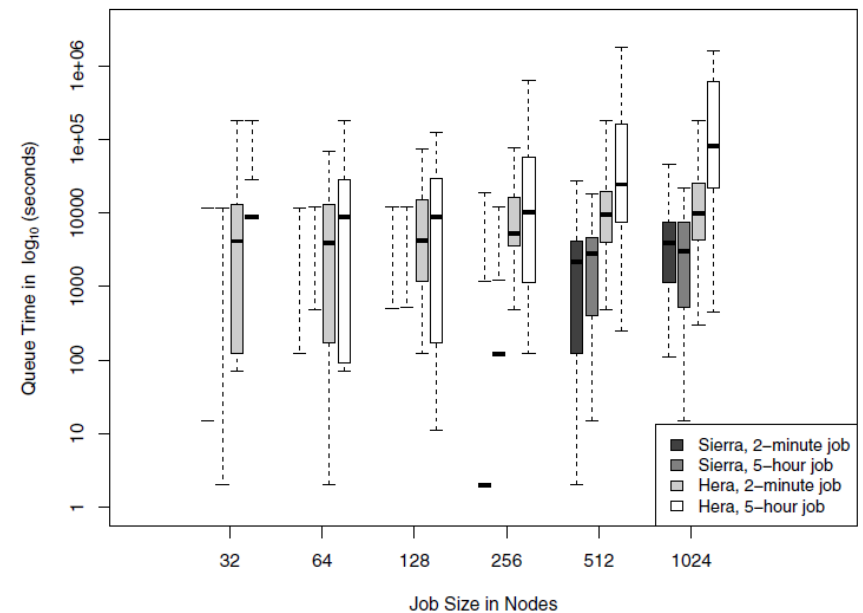
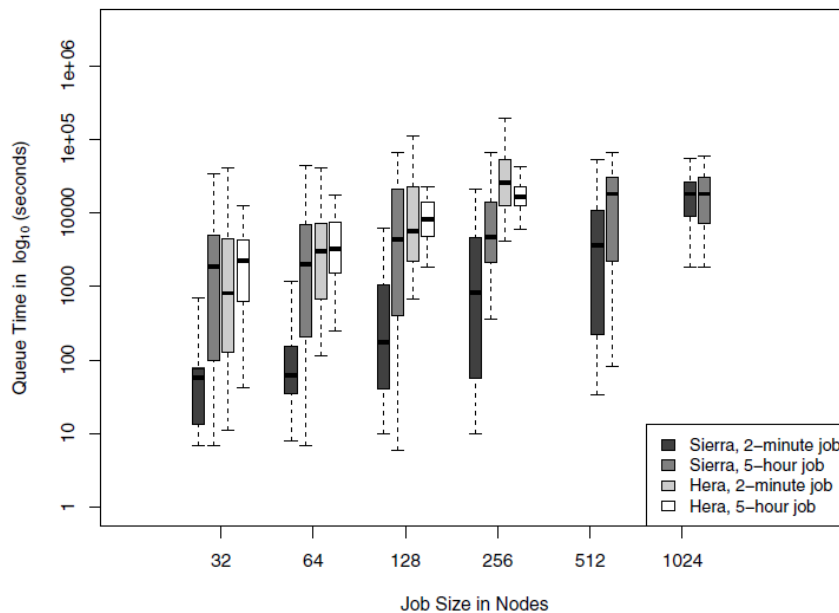
---

- ▶ Wait times are thus simulated by a simulator which is configured to work like the clusters
- ▶ Simulator used job logs from a different cluster taken in 2009 to simulate a proper workload. This workload has characteristics similar to the test clusters.
- ▶ To test the validity of the simulator, real queue wait times were collected for a few test jobs over a two-month period.



# Queue Wait Times Cont.

- ▶ The simulated times follow the same trend as the real wait times.
- ▶ Note that EC2 wait times were orders of magnitude smaller, with the highest being 244 seconds for 64 nodes.



# Total Turnaround Time

---

- ▶ Now that the simulation gives a reliable distribution of queue wait times, it can be used to determine the total turnaround time relative to EC2.
  - ▶ Remember, EC2 wait times are *considerably* lower than the HPC clusters because of the cluster's queuing.
- ▶ Tested using an MPI task set to run on 5 hours on EC2. Results are scaled relative to EC2's run time.
- ▶ Tested with 256, 512, and 1024 tasks.



# Total Turnaround Time Trends

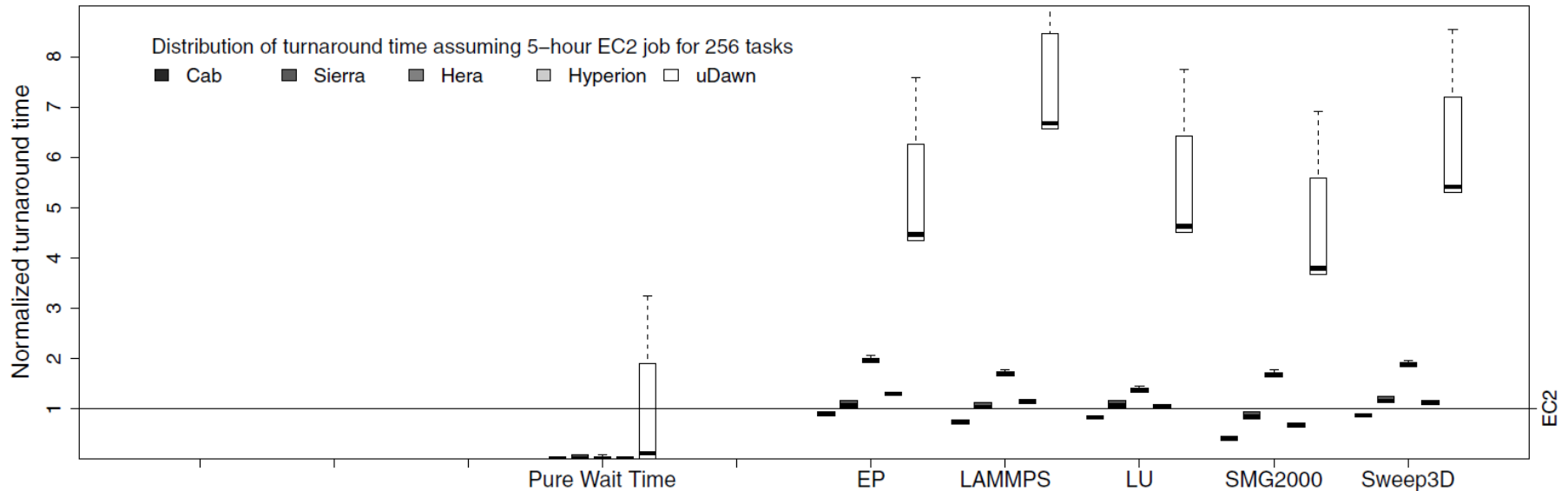
---

- ▶ EC2 *execution* time is generally better at lower scales.
- ▶ However, as the number of tasks increases, EC2 is outstripped by the better-scaling high end LLNL clusters. This makes sense, as EC2 has weaker communication.
- ▶ Turnaround time for the LLNL clusters is highly dependent on the queue wait time, as a higher task count causes longer turnaround times.
- ▶ This dependency can be seen by the fact that 1<sup>st</sup> quartile (best case) queue wait times make LLNL clusters have the better turnaround time, but 4<sup>th</sup> quartile (worst case) queue times give EC2 the advantage.



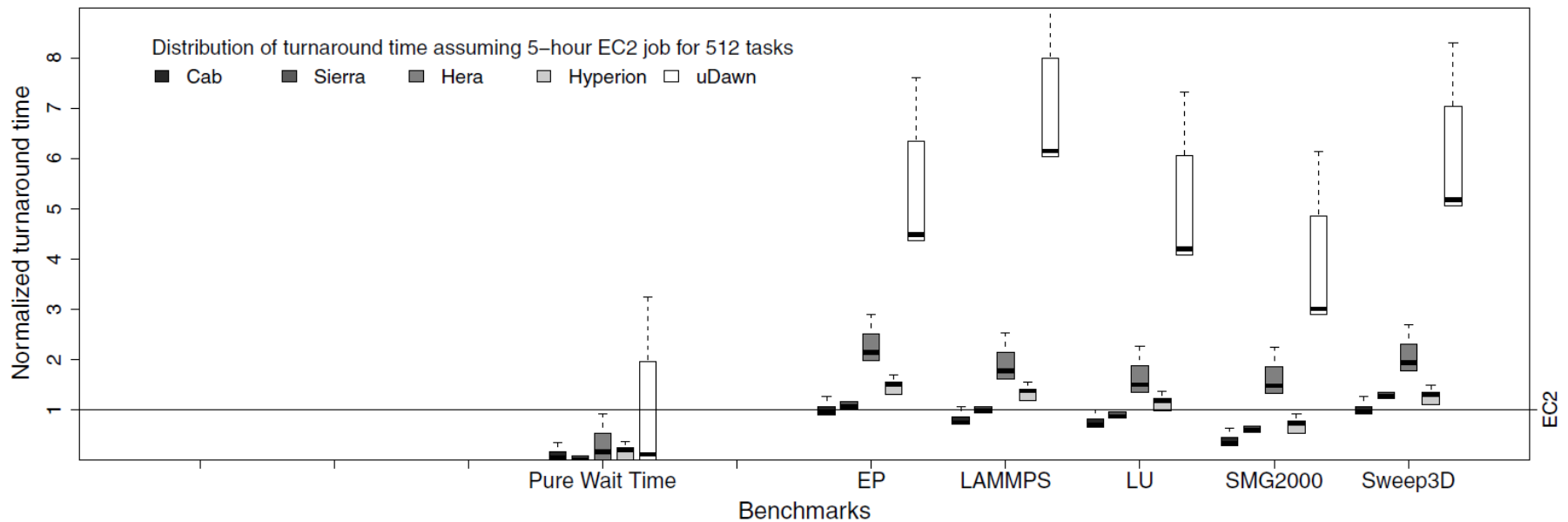
# Graphical Turnaround Time 256 Tasks

- ▶ Notice that EC2 beats many of the dedicated clusters for low task counts



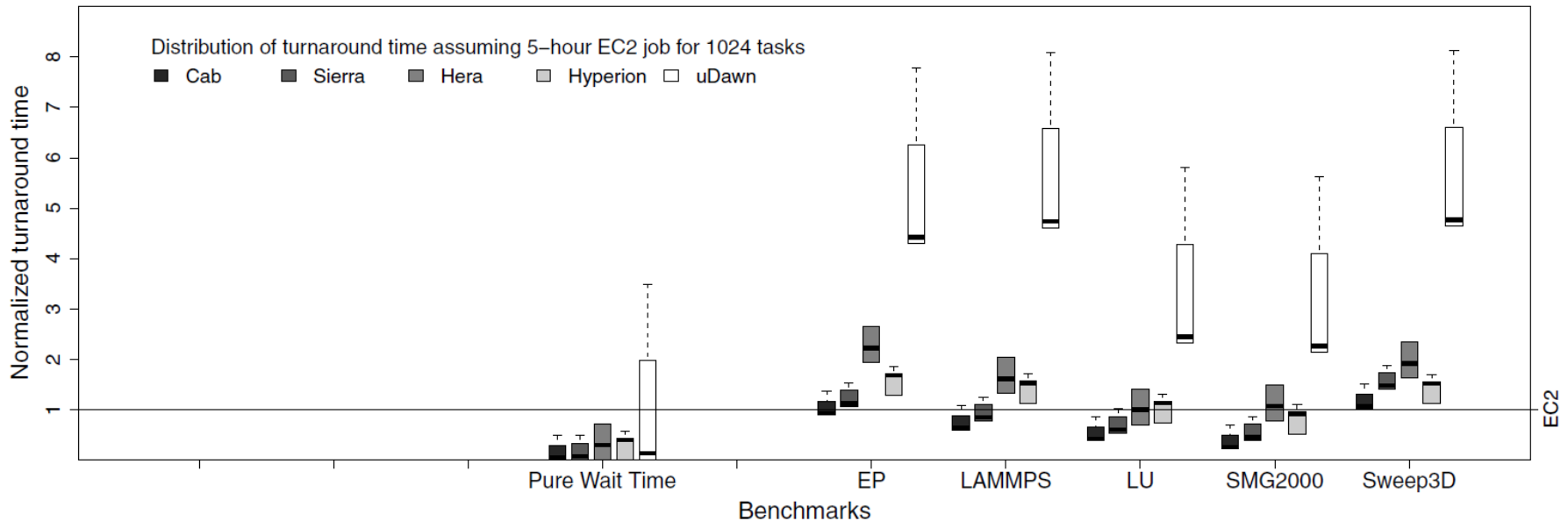
# Graphical Turnaround Time 512 Tasks

- ▶ Here, we see that machines like Cab and Sierra are sometimes right on the line with EC2, which means that the queue wait time will decide which is better.



# Graphical Turnaround Time 1024 Tasks

- ▶ Here, we see that in many cases, the better turnaround time all depends on the queue wait time, however Cab and Sierra outright beat EC2 for LU and SMG2000, while EC2 still beats them all in Sweep3D



## Expected Amount of Time LLNL is Better

---

- ▶ Using QBETS (a time series estimation method), it can be determined how often LLNL beats EC2 in terms of total turnaround time.
  - ▶ Only done for 1024 tasks
- ▶ The 0% instances mean EC2 is always better.
- ▶ Even with the fastest machine, the expectation ranges from 25-40%. This is further proof that the queue wait times play a significant factor in total turnaround time.

Application	Cab	Sierra	Hyperion	Hera	uDawn
EP	25%	0%	0%	0%	0%
Sweep3D	0%	0%	0%	0%	0%
LU	40%	35%	15%	25%	0%
SMG2000	35%	0%	0%	0%	0%
LAMMPS	35%	25%	0%	0%	0%





# Cost Calculation

---

- ▶ The price of EC2 is known, but the LLNL clusters need a price assigned to them.
- ▶ Calculated prices are based off of EC2 (\$2.40/node-hour)
- ▶ Can't view computation as a utility, because operating cost data for clusters isn't publicly available.
- ▶ Assume LLNL clusters are competitors in the same market as EC2
- ▶ Do not incorporate queue wait times in the cost.
- ▶ Users in this market make “optimal” choices for profit maximization.



# Cost Calculation Cont.

---

- ▶ User takes into account:
  - ▶  $P$  = price in node-hours
  - ▶  $N$  = nodes to allocate
- ▶ User wants to minimize the combined implicit and explicit costs:
  - ▶  $C = (p * n * t(n)) + (a * t(n))$
- ▶ The first term is the explicit cost; it is the actual monetary value to be spent on calculation.
- ▶ The second term is the implicit cost; this can be the cost of waiting longer for something slow to finish, etc.



## Cost Calculation Cont.

---

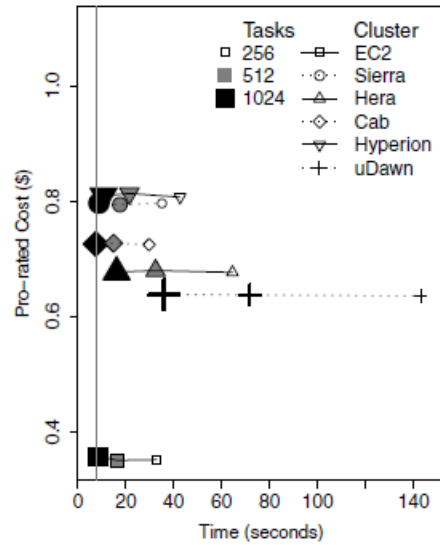
- ▶ Using this model of user choice, cluster operators can choose a price which can maximize profit.
- ▶ Lots of math later, and we get the following prices:

CC2	Hera	Sierra	Hyperion	uDawn	Cab
2.40	2.36	3.83	2.13	0.25	5.49

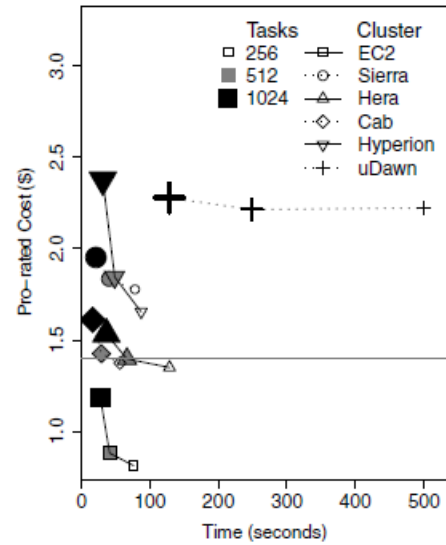
- ▶ Notice that the slowest machine, uDawn, has a very low price, while the best machine, Cab, has a fairly high price.
- ▶ Note that EC2 uses hourly pricing, but in order to gather data more effectively for the short jobs that were run, the prices were prorated by the second for the following data.



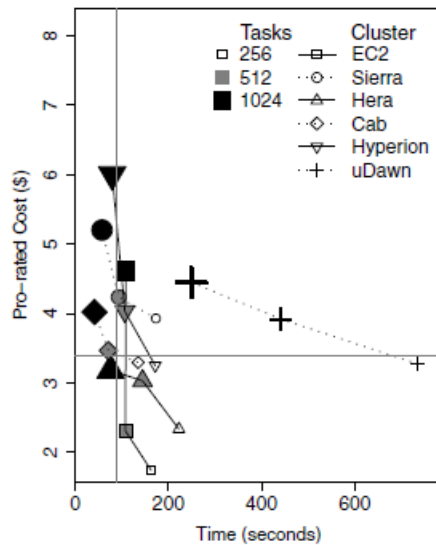
Cost/Time Tradeoff: EP



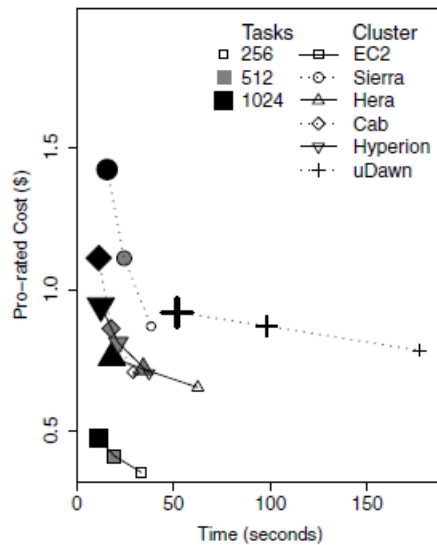
Cost/Time Tradeoff: LAMMPS



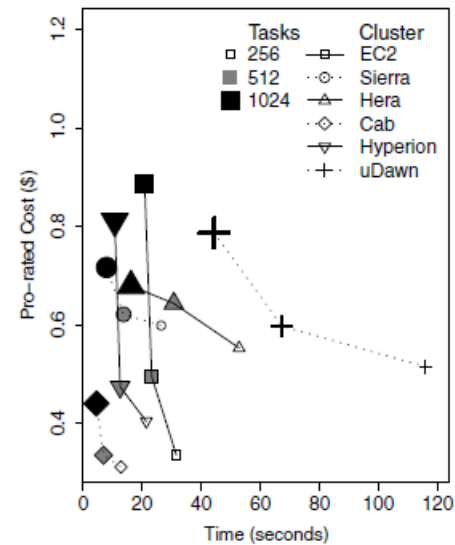
Cost/Time Tradeoff: LU



Cost/Time Tradeoff: Sweep3D



Cost/Time Tradeoff: SMG2000



# What Do These Results Show?

---

- ▶ Different applications have their total cost minimized on different clusters. In other words, a single cluster (such as the “cheap” uDawn) doesn’t always have the lowest cost.
  - ▶ For instance, it is cheapest to run computationally intensive but communication-lacking programs on EC2, as EC2’s price is optimized for computation.
  - ▶ Alternatively, it is cheapest to run communication-intense programs on Cab since Cab’s price is optimized for communication.
- ▶ As we saw before, different applications have their shortest running time on different clusters
  - ▶ It all depends on the qualities of the application: is it cache heavy? Communication-intense? Computationally-intense?



# More Results

---

- ▶ If a program must be completed within a certain time, it can yet again change the machine which provides the lowest cost.
- ▶ Similarly, if a price bound exists, the machine which provides the fastest time may change
- ▶ **Basically:** many factors affect the choice of machine:
  - ▶ Queue wait times
  - ▶ The application itself
  - ▶ Whether turnaround time or cost is more important
  - ▶ Time/cost bounds



# Well, Which One To Choose?

---

- ▶ The choice between HPC platforms isn't clear; users cannot exhaustively test their application on a multitude of clusters/cloud providers to determine which one is best for their needs.
- ▶ Authors propose software that can choose a cluster automatically.
  - ▶ This would require that systems provide wait queue data, which could be a security concern.



# Bottom Line

---

- ▶ The cloud isn't entirely useless for HPC problems. It outperforms dedicated HPC hardware in some cases.
- ▶ When measured in total turnaround time, the cloud can outperform dedicated HPC hardware in many cases due to significant queue wait time on HPC clusters.
- ▶ Should current HPC cluster providers start charging for computation in a manner similar to cloud providers, a multitude of factors can affect a user's machine choice. The choice is not clear in this case, and the responsibility of navigating these factors shouldn't be placed on the user.

