Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Automated benchmarking for resource provisioning

Shrijan Swaminathan, Marco Mambelli

Throughput Computing 2023

13 July 2023

DOE Omni Technology Alliance INTERNSHIP PROGRAM
de omnibus dubitandum
Be suspicious of everything

PURDUE UNIVERSITY
College of Engineering

Why should we benchmark?

- Worker nodes are all different from each other.
  - It is hard to compare due to differing architecture.
  - Different ratio between CPU cores and RAM.

- To evaluate the total performance of a machine in such a way that it is easily comparable but also while minimizing error (in terms of oversimplification).

- Better quality nodes can reduce costs on clouds and optimize cloud VM purchases, which transfers to higher throughput overall.

Manual benchmarks

I mainly used the HepSpec06 (HS06) benchmark to run tests on different types of cloud renting virtual machines. I primarily used AWS EC2 and Google Cloud Compute as a basis for the testing.

The main reason for running the benchmarks was to get price to performance data of instances in order to figure out what the most efficient instances to purchase based on hardware performance.

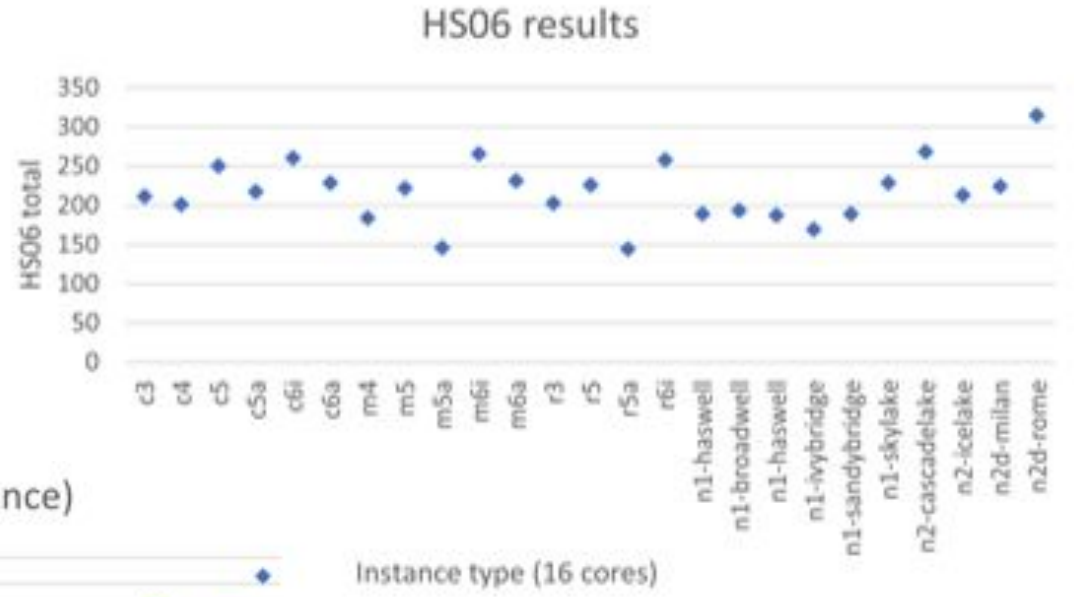
Manual benchmark results

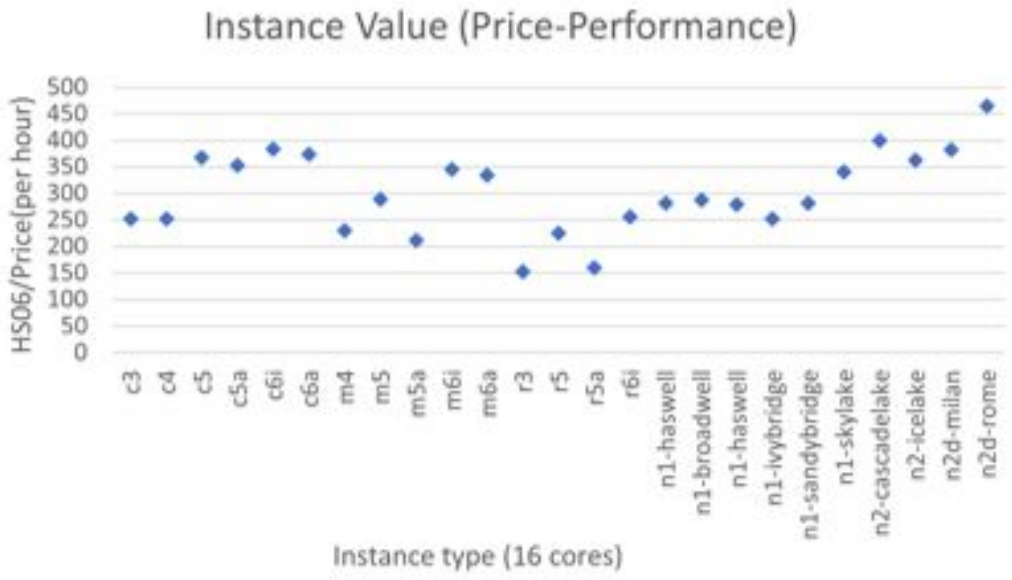
Amazon	N_COR	CORE TYPE	Speed(GHz	$ per hou	HS06 per cor	HS06 tota	HS06 per $/
c3	16	Xeon E5-2680	2.80	0.840	13.2	212	252
c4	16	Xeon E5-2666	2.90	0.796	12.6	201	252
c5	16	Xeon Platinum 8275CL	3.00	0.680	15.6	250	368
c5a	16	AMD EPYC 7R32	3.3*	0.616	13.6	217	353
c6i	16	Xeon Platinum 8375C	2.90	0.680	16.3	260	383
c6a	16	AMD EPYC 7R13	2.0*	0.612	14.3	229	374
m4	16	Xeon E5-2686	2.30	0.800	11.5	184	229
m5	16	Xeon Platinum 8259CL	2.50	0.768	13.9	222	289
m5a	16	AMD EPYC 7571	2.1*	0.688	9.1	146	212
m6i	16	Xeon Platinum 8375C	2.90	0.768	16.6	266	346
m6a	16	AMD EPYC 7R13	2.0*	0.691	14.4	231	334
r3	16	Xeon E5-2670	2.50	1.330	12.7	203	153
r5	16	Xeon Platinum 8259CL	2.50	1.008	14.2	226	225
r5a	16	AMD EPYC 7571	2.1*	0.904	9.0	144	160
r6i	16	Xeon Platinum 8375C	2.90	1.008	16.1	258	256

Google	N_COR	CORE TYPE	Speed(GHz	$ per hou	HS06 per cor	HS06 tota	HS06 per $/
n1-broadwell	16	Intel Xeon CPU	2.20	0.672	11.8	189	281
n1-haswell	16	Intel Xeon CPU	2.30	0.672	12.1	193	287
n1-ivybridge	16	Intel Xeon CPU	2.50	0.672	11.7	187	279
n1-sandybridge	16	Intel Xeon CPU	2.60	0.672	10.6	169	252
n1-skylake	16	Intel Xeon CPU	2.00	0.672	11.8	189	281
n2-cascadelake	16	Intel Xeon CPU	2.80	0.672	14.3	229	340
n2-icelake	16	Intel Xeon CPU	2.60	0.672	16.8	268	399
n2d-milan	16	AMD EPYC 7B13	2.50	0.587	13.3	213	363
n2d-rome	16	AMD EPYC 7B12	2.25	0.587	14.0	224	382
t2d-milan	16	AMD EPYC 7B13	2.50	0.677	19.7	315	465

Manual benchmark results (contd.)

Raw performance (right) is higher for higher generation cpus.





Even more pronounced for the performance over price ratio (left).

On-demand vs Spot pricing

Cloud VM pricing is split into two types: On-demand and Spot

On-demand

- Fixed pricing
- Can run work for as long as the user wants
- Expensive (in comparison to Spot)
- Works similar to a service purchase

Spot

- Varied pricing
- Has the possibility for being pre-empted
- Can be much cheaper than on-demand pricing
- Works similar to an auction

Using spot pricing reduces cost, which reduces price-performance as a whole.

Use of Benchmark Results

HEPCloud's Decision Engine at Fermilab uses a spot pricing algorithm to provision cloud resources.

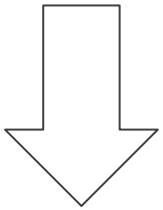
It uses the values from the benchmarks to calculate the figure of merit of each VM type and minimize the cost.

Manual vs Automated Benchmarking

- Many different new platforms are added all the time.
- Tailored benchmarks could optimize for different types of user jobs.

Manual vs Automated Benchmarking

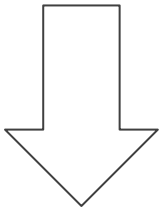
- Many different new platforms are added all the time.
- Tailored benchmarks could optimize for different types of user jobs.



- We need many benchmarks.

Manual vs Automated Benchmarking

- Many different new platforms are added all the time.
- Tailored benchmarks could optimize for different types of user jobs.

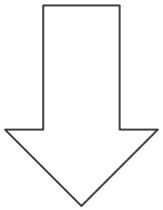


- We need many benchmarks.

- Manual Benchmarking is labor-intensive.

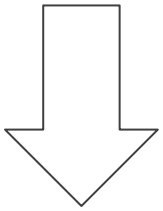
Manual vs Automated Benchmarking

- Many different new platforms are added all the time.
- Tailored benchmarks could optimize for different types of user jobs.



- We need many benchmarks.

- Manual Benchmarking is labor-intensive.

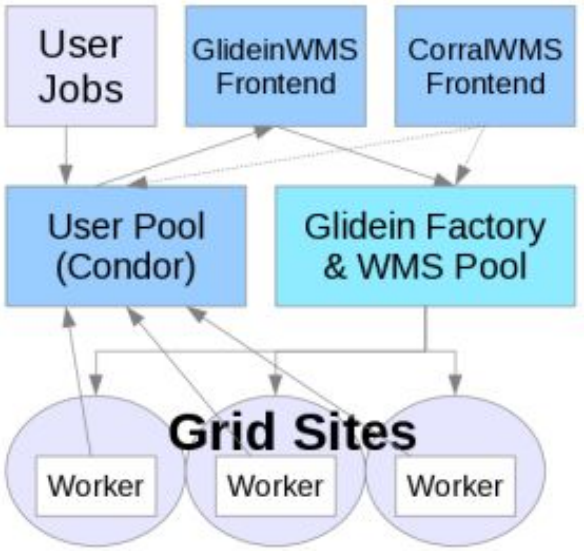


We need automation!

Automated Benchmarking via Glideins

GlideinWMS is a workload management system, which uses HTCondor to send Glideins and provision resources.

We want to leverage Glideins to run the benchmarks.



Manual vs GWMS Automated Benchmarking

Manual

- Labor-intensive
- Can run as root
- Run on barebone hardware or VM
- Easier to customize
- Limited resources
- Controlled environment

Automated via GWMS

- Easier to run
- Unprivileged only
- Run as container
- Standard versions
- Easier to scale
- Run on provided resource

Running benchmarks in Glideins

- Containers provide a standard environment.
- Apptainer can run unprivileged.
- CVMFS can provide multiple benchmarks in expanded images.
- Find root-less benchmarks.

```
#!/bin/sh

glidein_config="$1"

# find error reporting helper script
error_gen=`grep '^ERROR_GEN_PATH ' "$glidein_config" | awk '{print $2}'`

cd "$TMP"
OUTPUT_DIR="$TMP/atlasgenbmk"

if [ ! -d "$OUTPUT_DIR" ]; then
    echo "$OUTPUT_DIR" does not exist.  Trying to create it...
    if ! mkdir -p "$OUTPUT_DIR"; then
      "$error_gen" -error "atlasgenbmk.sh" "WN_Resource" "Could not create $OUTPUT_DIR"
      exit 1
    fi
fi

COUNT=`cat /proc/cpuinfo | grep -c processor`
singularity run -i -c -e -B "$OUTPUT_DIR":/results /cvmfs/unpacked.cern.ch/gitlab-registry.cern.ch/hep-benchmarks/hep-workloads/atlas-gen-bmk:v2.1 -W --threads $COUNT --events 100
#for future use, change the events to something else other than 10. 10 is just to test if the script works well under gwms

if [ -f "$OUTPUT_DIR/atlas-gen_summary_new.json" ]; then
   cat "$OUTPUT_DIR"/atlas-gen_summary_new.json
else
  "$error_gen" -error "atlasgenbmk.sh" "WN_Resource" "Could not find $OUTPUT_DIR/atlas-gen_summary_new.json"
   exit 1
fi
"$error_gen" -ok "atlasgenbmk.sh"
exit 0
```

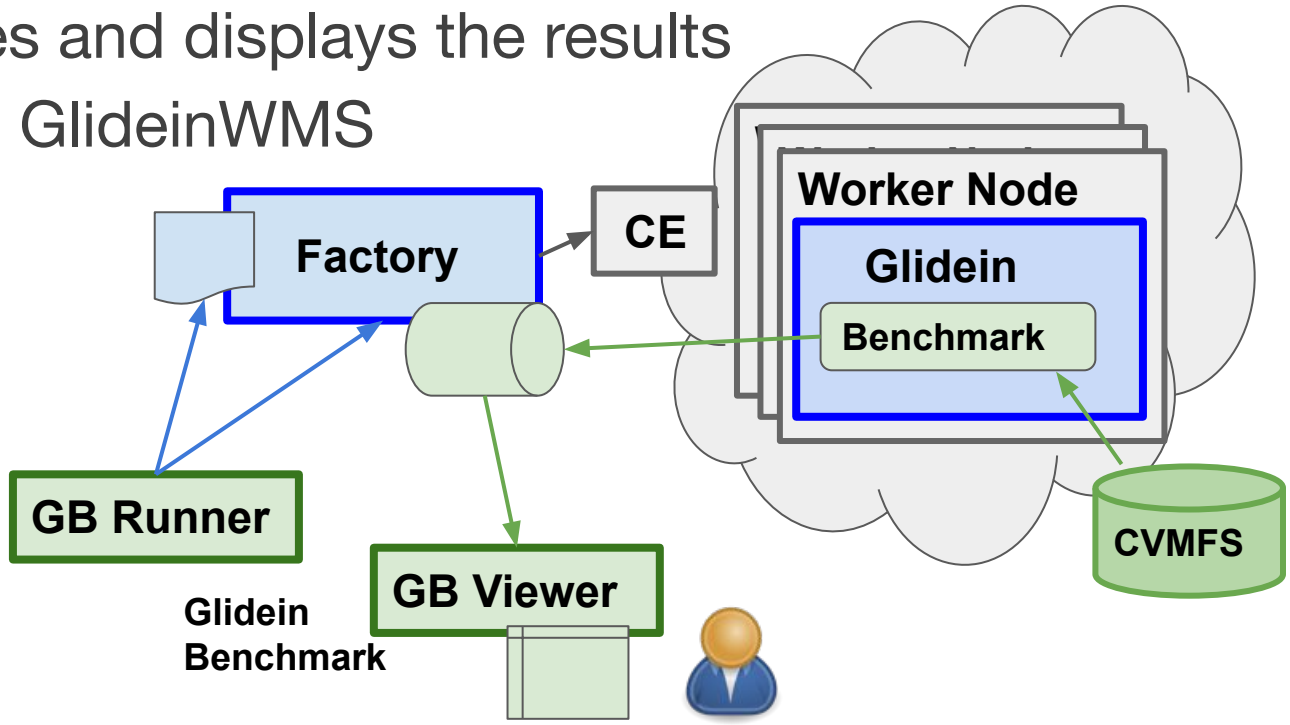
GlideinBenchmark

Web application/dashboard to automate benchmarks via GlideinWMS.

Two components:

- the runner triggers and monitors the benchmark execution
- the viewer serves and displays the results

Works with existing GlideinWMS deployments

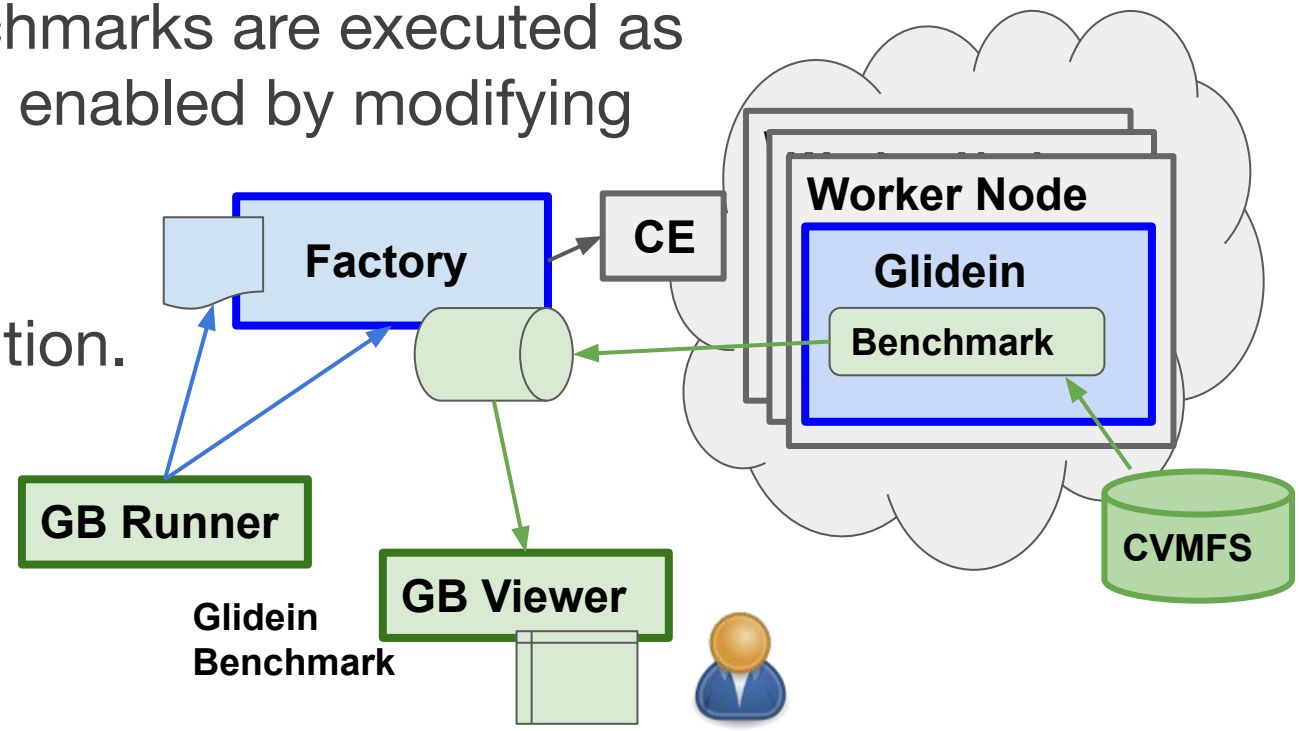


Runner

Executes on the desired resources the benchmark selected by the user by controlling the GWMS Factory:

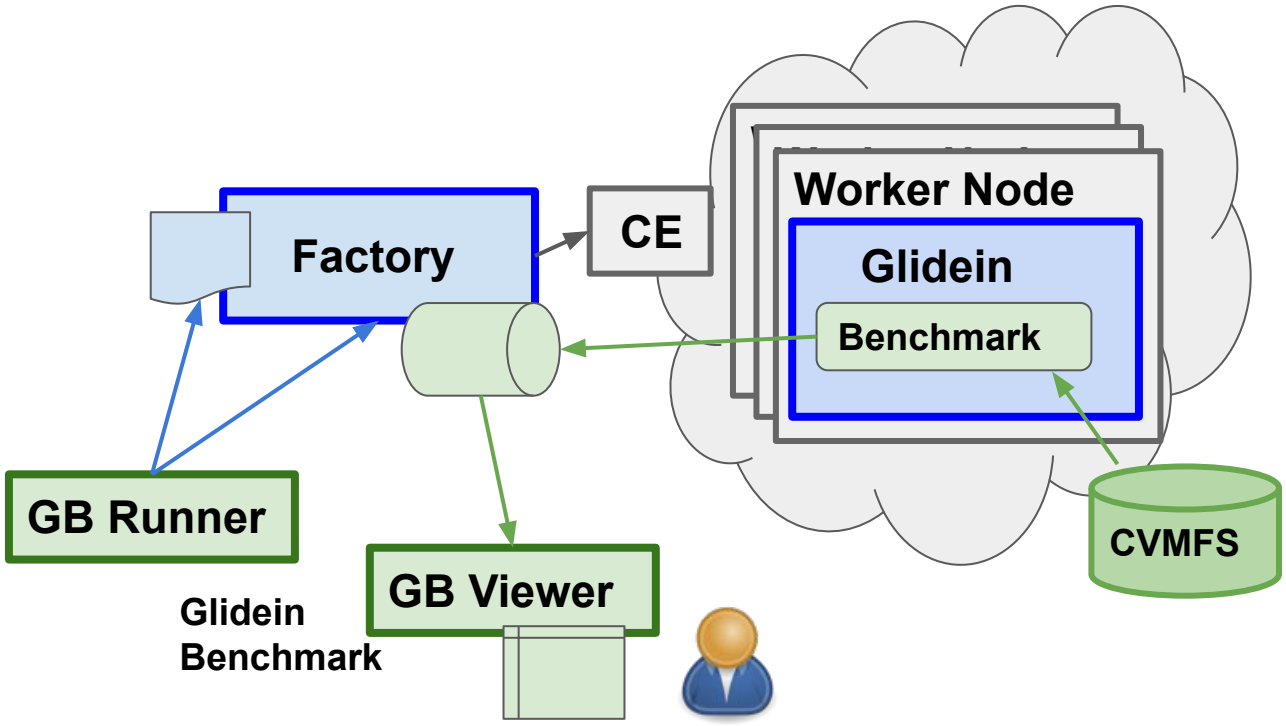
- On-demand benchmarks are triggered via HTCondor job submissions.
- Automatic benchmarks are executed as Glidein test and enabled by modifying the Factory configuration.

Monitors the execution.



Viewer

Web dashboard serving via a RESTful interface and displaying the benchmark results.



Thank you and Acknowledgements

This work was done under the DOE Omni Technology Alliance Internship Program by ORISE

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

Thank you Throughput Computing 23 for hosting such an amazing event and for facilitating my attendance.

Summary

- Benchmarking overall is a very useful tool in summarizing performance of a machine and to optimize resource provisioning.

- Automated benchmarking is needed to run updated benchmarks on all the resources.

- GlideinBenchmark is a web application I'm developing that leverages Glideins and HTCondor to submit customized benchmarks. Experiments can use it to rank the resources and reduce their execution costs.