Metrics & Measurements

WLCG System Performance Working Group.
Gareth Roy (University of Glasgow)
Markus Schulz (CERN)
Andrea Sciaba (CERN)
Graeme Stewart (CERN)
Michel Jouvin (IN2P3)

Outline

- Introduction
- Resource view
- Metrics
- Measurements
- Conclusion

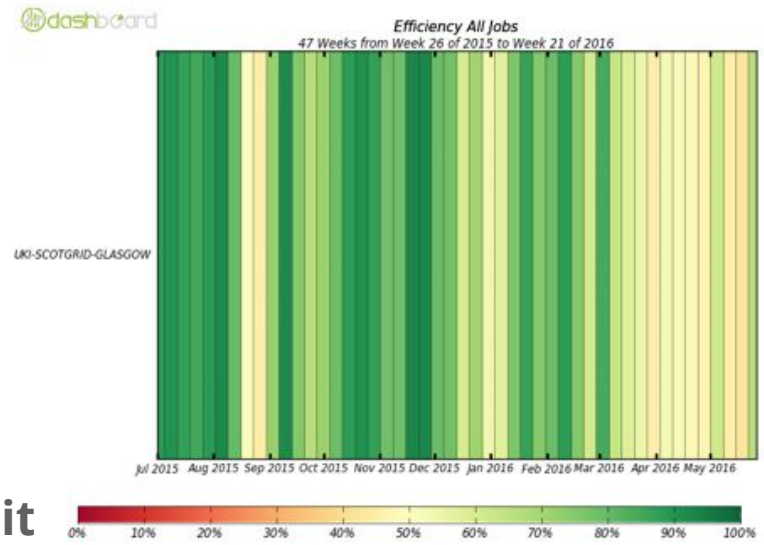
Introduction

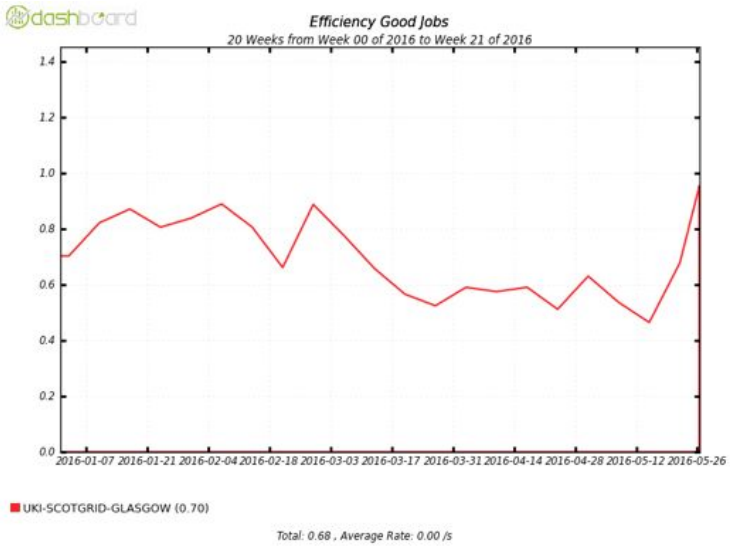
Our community needs **metrics** that allow us to **characterise** the resource usages of HEP workloads in sufficient detail so that the **impact of changes** in the infrastructure or the workload implementations can be quantified with a precision high enough to **guide** design decisions towards improved efficiencies. This model has to express the resource utilisation of the workloads in terms of fundamental capabilities that computing systems provide, such as storage, memory, network, computational operations, latency, bandwidths etc.

https://twiki.cern.ch/twiki/bin/view/LCG/WLCGSystemsPerformanceModeling

Impact of Changes

- From mid 2015 to mid 2016 UKI-SCOTGRID-GLASGOW observed an approx. 45% drop in job efficiency for ATLAS reconstruction workflows.
- Investigation showed many WN disks maxed out on total number of IOPS.
- This appeared as a significant amount of **iowait** as jobs stalled waiting to read data from disk.
- The causes were identified as:.
  - Job Mix: MC Production had dried up at the site, with nearly all payloads being Reconstruction.
  - Increased payload size: Staged sandboxes increased from 20GB to 50GB in size.
- A change in usage patterns had led to a huge drop in efficiency. Compute resource were not spec'd to handle that amount of I/O.



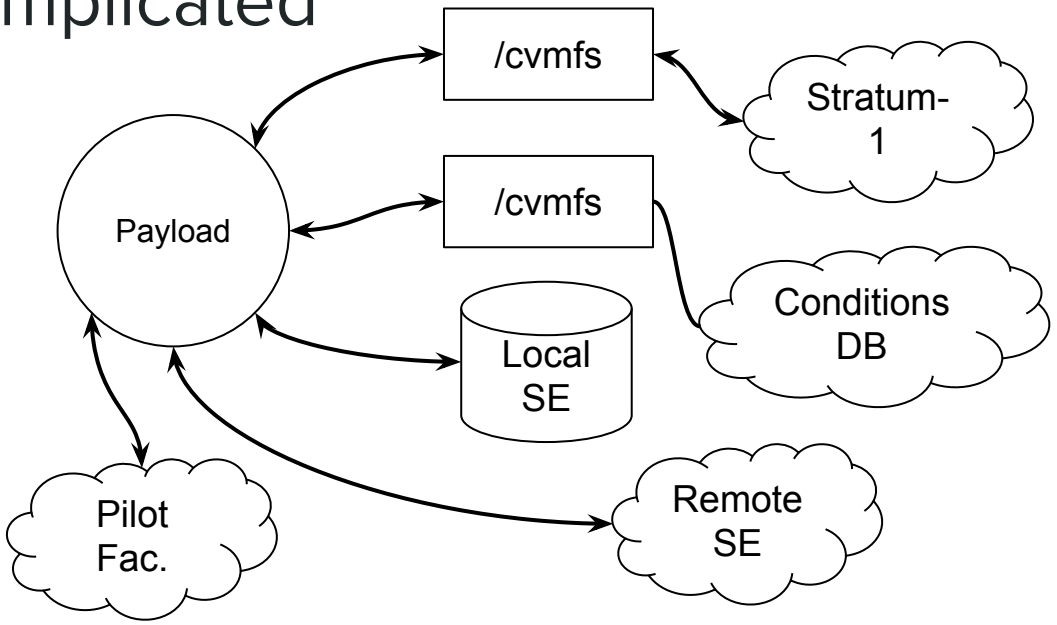


WLCG Payloads are complicated

Unlike many other communities (such as HPC) WLCG payloads are complex due to the distributed nature of the Grid.

- Software distributed via **/cvmfs**.
- Datasets are obtained from Local Storage Elements (SE) via third party copies.
- Some experiments access data from remote SE's via the WAN.
- Some simulation information (such as Condition data) may be stored via **/cvmfs** or accessed remotely via FRONTIER.
- Each job has multiple monitoring paths, communicating with local batch and remote pilot systems.

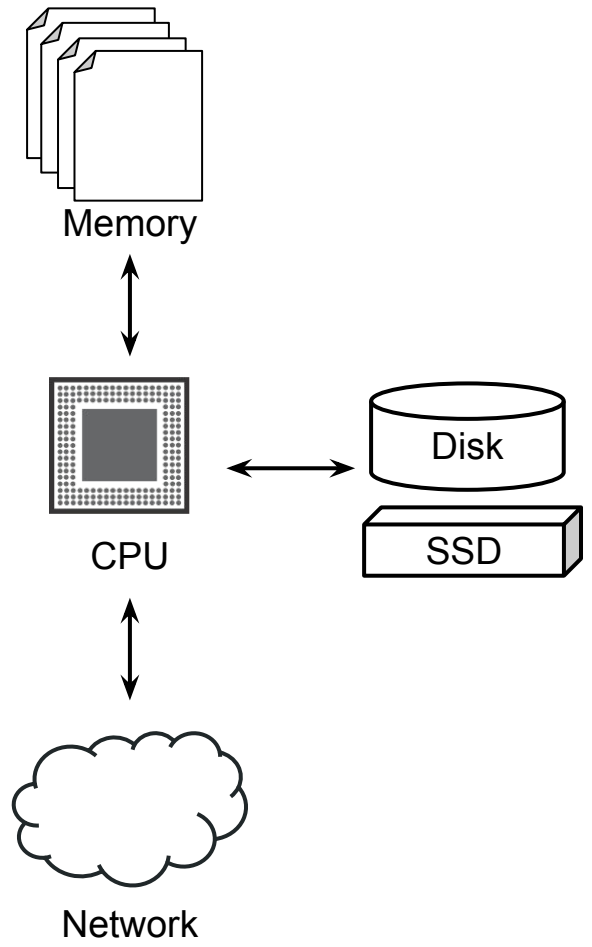
It is difficult to assess the performance impact of all these moving parts.




```
Aug18   0:08  |   \_ /usr/bin/docker-containerd-shim-current a8d6581f1facdd718bb8349af7c95b4771056a3d42b27
Aug18   0:00  |   |   \_ /bin/bash /cntwn.sh
Aug18   0:03  |   |       \_ condor_master -pidfile /var/run/condor/condor_master.pid
Aug18   0:28  |   |       |   \_ condor_procd -A /var/run/condor/procd_pipe -L /var/log/condor/ProcLog -R
Aug18   0:12  |   |       |   \_ condor_shared_port -f -p 9621
Aug18   3:40  |   |       |   \_ condor_startd -f
14:53   0:01  |   |       |       \_ condor_starter -f -a slot1_1 svr019.beowulf.cluster
14:53   0:00  |   |       |           \_ /bin/bash -l /var/lib/condor/execute/dir_30397/condor_exec.exe
14:53   0:00  |   |       |               \_ /usr/bin/time -o /var/lib/condor/execute/dir_30397/cP4MDm21S3
14:53   0:00  |   |       |                   \_ /bin/bash ./runpilot3-wrapper.sh -s UKI-SCOTGRID-GLASGOW_
14:53   0:07  |   |       |                       \_ /usr/bin/python pilot.py -d /var/lib/condor/execute/d
14:53   0:06  |   |       |                           \_ /usr/bin/python RunJob.py -a /cvmfs/atlas.cern.ch
14:56   0:00  |   |       |                               \_ /bin/sh -c export ATHENA_PROC_NUMBER=8;export
14:56   0:29  |   |       |                               |   \_ python /cvmfs/atlas.cern.ch/repo/sw/softw
16:17   0:00  |   |       |                               |       \_ /bin/sh ./runwrapper.ESDtoAOD.sh
16:17   0:47  |   |       |                               |       |   \_ /cvmfs/atlas.cern.ch/repo/sw/soft
16:19   6:16  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:17  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:14  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:14  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:13  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:15  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:18  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:19   6:13  |   |       |                               |       |       \_ /cvmfs/atlas.cern.ch/repo/sw/
16:17   0:07  |   |       |                               |       \_ MemoryMonitor --pid 16780 --filename
14:56   0:00  |   |       |                               \_ /bin/sh -c cd /var/lib/condor/execute/dir_303
14:56   0:49  |   |       |                                   \_ MemoryMonitor --pid 12170 --filename memo
Aug18   0:02  |   |       \_ tail -f /var/log/condor/KernelTuning.log /var/log/condor/MasterLog /var/log/c
```

A simplified resource view of a Payload

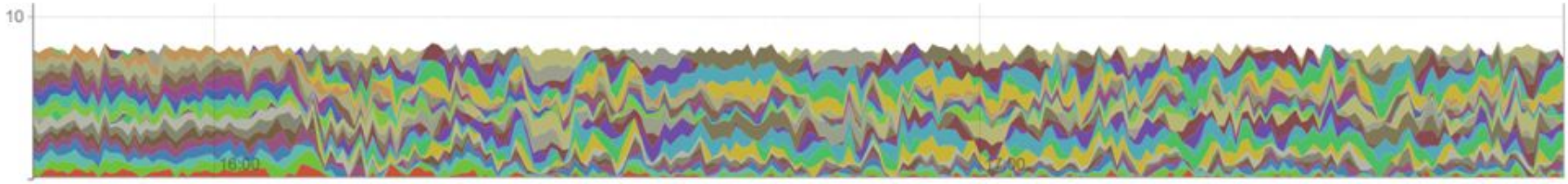
- We need a simplified view of resource utilisation to measure appropriately.
- It is important to identify metrics & tools that allow us to characterise a payload in the terms of the four primary system resources:
  - CPU
  - Memory
  - Disk I/O (i.e. file, block device etc.)
  - Network I/O (staging, etc.)
- Resource metrics can be combined with application level metrics such as:
  - Input, output dataset sizes
  - Throughput such as evnt/s etc.
- A simple set of metrics can be obtained to begin exploration of system performance.



Sample Metrics (CPU)

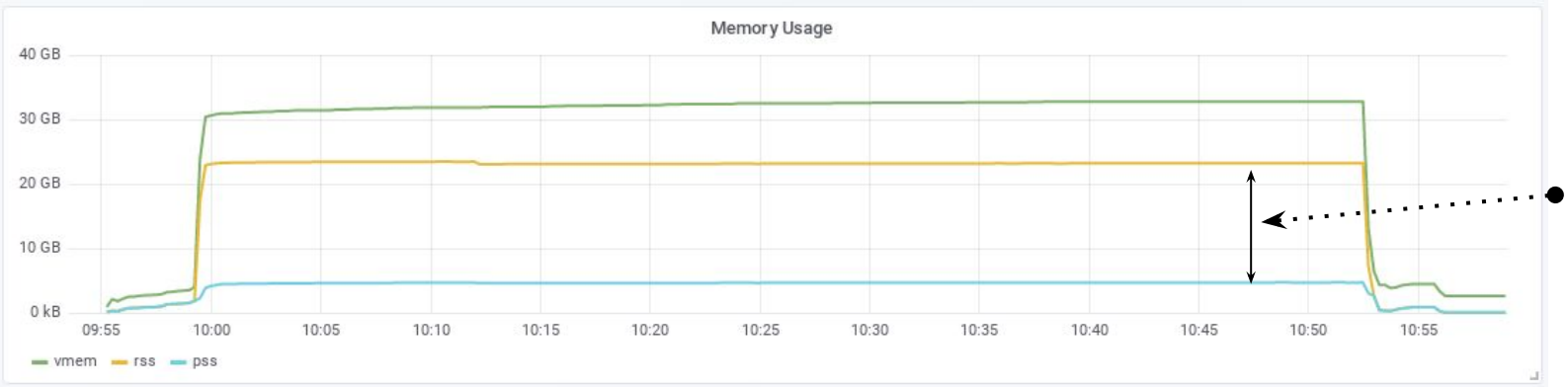
Metric	Type	Source	Scope	Command	Insight	Comments
%usage	gauge	Tool internal	process	/bin/time <x> prmon	Gross measure of cpu utilisation, real/user/sys. Indicates potential overheads and multi-process scaling.	Use application metric of event loop time to change all of these per second metrics into per event (see below)
Thread #	gauge	/proc/<pid>/status	process	grep Threads	Gives a measure of how much of a running payload is parallel/serial.	Required for multi-threaded code
Process #	gauge	Process list	process	pstree -p <p> |wc	As above but for multi-process codebases.	Required for multi-process code

- Initial attempt to build a table of metrics and the insight we can gain from each metric.
- Identify data sources and simple commands that can be used to extract that information.
- Still a work-in-progress and will be expanded as we start to measure real world payloads
    - For instance %usage misses context switches between individual CPU cores (graph below) of running payloads. Need to capture cpu counters, instructions/cycle etc. to get a full picture.



Sample Metrics (Memory)

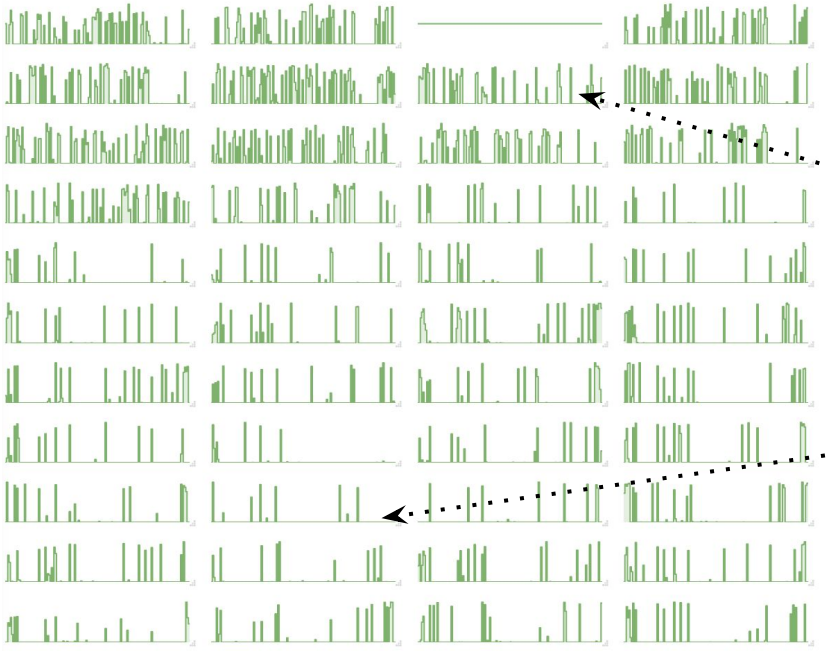
Metric	Type	Source	Scope	Command	Insight	Comments
Memory usage	gauge	/proc/<pid>/smaps /proc/<pid>/status	process	prmon	Allows understanding of how memory develops over time, can be used in conjunction with Process/Thread count to examine dependency.	*VMEM is application controlled, RSS is how much the kernel really maps, PSS accounts for shared pages better (important for parallel processing).*
Avg Mem	gauge	/proc/<pid>/smaps	process	prmon	Amount of memory that needs budgeted for the bulk of the runtime of the job payload.	*(see above)*
Max Mem	gauge	/proc/<pid>/smaps	process	prmon	Amount of memory that needs to be made available instantaneously - required for setting hard limits on a job payload to detect erroneous jobs.	(see above)



RSS reports 20GB more memory usage than is actually consumed when using shared pages

Sample Metrics (IO)

Metric	Type	Source	Scope	Command	Insight	Comments
I/O rate	gauge	/proc/diskstats	global	iostat 1 1	Total IO operations ongoing, can calculate a %usage of theoretical maximum of spinning/ssd media	As /proc/diskstats is global some method of isolating a process is necessary to assess accurately (containers/namespaces?)
I/O bandwidth	gauge	/proc/<pid>/io	process	prmon	Total bytes read/written by a process, gives indication of rates and total usage	

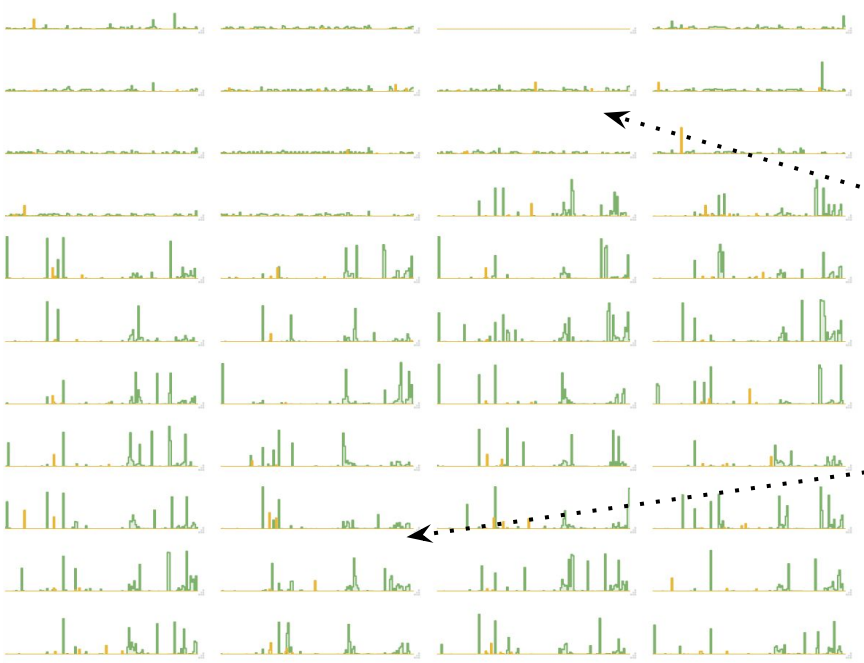


Sparklines showing I/O rates (bytes/s) for 44 VAC hosts.

- 14 hosts running single core LHCb workloads
- 30 hosts running 8 core ATLAS Production Payloads

Sample Metrics (Network)

Metric	Type	Source	Scope	Command	Insight	Comments
Network usage	gauge	/proc/net/dev	global	Possible update to prmon	Aggregate Tx/Rx bytes to assess total network load	As /proc/net/dev is global some method of isolating a process is necessary to assess accurately (containers/namespaces?)
Network rates	gauge	Socket statistics	process	ss -ip	Per process rates, can be used to assess /cvmfs usage.	More work needed to understand if the numbers provided are useful



Sparklines showing network usage for 44 VAC hosts:

- 14 hosts running single core LHCb workloads
- 30 hosts running 8 core ATLAS Production Payloads

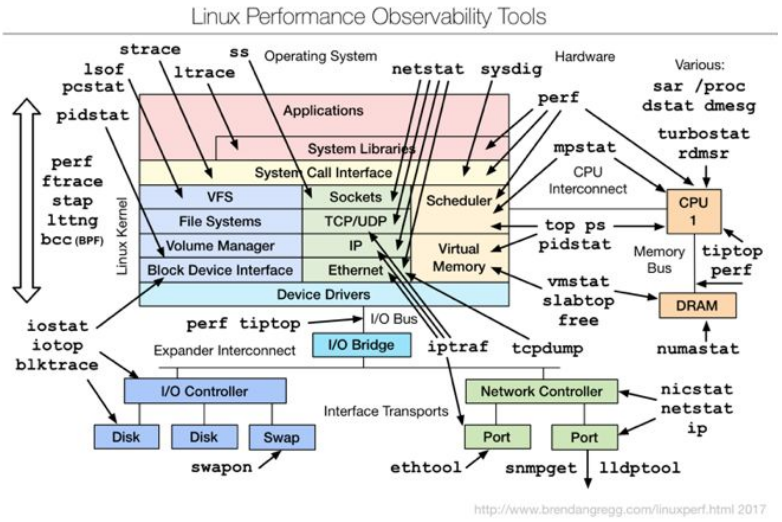
Sample Metrics (Application)

Metric	Type	Source	Scope	Command	Insight	Comments
Input data set	gauge	Job payload	job	ls -lh	Size of input data set, space on disk required and total size to be staged.	This changes if data is streamed via remote IO and is not as simple as the size of all the files associated with a job.
Output data set	gauge	Job stageout	job	ls -lh	Size of output data set, amount of data that needs to be staged out	
Non Event input Data	gauge	Conditions DB etc.	global	-	Data overheads that are not directly related to the input data set but required.	Difficult to measure in a generic way

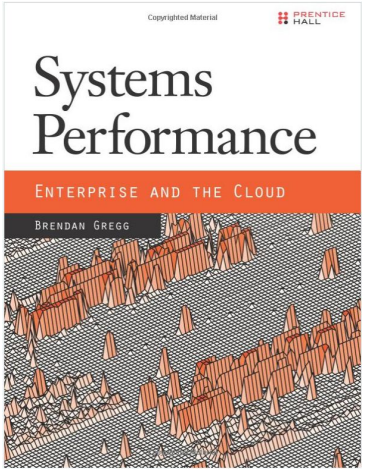
```
[root@kube003 rundir]# ls -ltrLh
total 27G
-rw-------.  1 root root 4.6G Mar 21 09:56 data17_13TeV.00327862.physics_Main.merge.AOD.f836_m1824._lb0168._0003.1
-rw-------.  1 root root 5.6G Mar 21 09:57 data17_13TeV.00327862.physics_Main.merge.AOD.f836_m1824._lb0169._0001.1
-rw-------.  1 root root 6.0G Mar 21 09:58 data17_13TeV.00327862.physics_Main.merge.AOD.f836_m1824._lb0169._0002.1
-rw-------.  1 root root 4.6G Mar 21 09:59 data17_13TeV.00327862.physics_Main.merge.AOD.f836_m1824._lb0169._0003.1
-rw-------.  1 root root 5.6G Mar 21 10:01 data17_13TeV.00327862.physics_Main.merge.AOD.f836_m1824._lb0170._0001.1
-rw-------.  1 root root 294M Mar 21 10:03 EVNT.13322104._000284.pool.root.1
-rwx--x--x.  1 root root 1.3K Mar 21 10:09 job_setup_mc.sh
-rwx--x--x.  1 root root 2.3K Mar 21 10:09 job_setup_reco.sh
-rwxr-xr-x.  1 root root 642K Mar 21 10:09 prmon
-rwxr-xr-x.  1 root root  10M Mar 21 10:09 beholder
-rw-r--r--.  1 root root 2.2K Mar 21 10:12 runargs.AODtoDAOD.py
-rwxr-xr-x.  1 root root  392 Mar 21 10:12 runwrapper.AODtoDAOD.sh
-rw-r--r--.  1 root root 3.7K Mar 21 10:13 athfile-6951-75d809e9-f321-404f-be80-4c9c62e23a95.log.txt
-rw-r--r--.  1 root root 143K Mar 21 10:13 athfile-cache.ascii.gz
-rw-r--r--.  1 root root  382 Mar 21 10:18 PoolFileCatalog.xml
drwxr-xr-x. 12 root root 4.0K Mar 21 10:18 athenaMP-workers-AODtoDAOD-a2da
-rw-r--r--.  1 root root 487M Mar 21 11:47 DAOD_HIGG8D1.13472023._000449.pool.root.1
-rw-r--r--.  1 root root   78 Mar 21 11:48 athenaMP-outputs-AODtoDAOD-a2da
-rw-r--r--.  1 root root 589K Mar 21 11:48 athenamp_eventorders.txt.AODtoDAOD
-rw-r--r--.  1 root root  15K Mar 21 11:48 mem.full.AODtoDAOD
-rw-r--r--.  1 root root 1.1K Mar 21 11:48 FileManagerLog
-rw-r--r--.  1 root root  71K Mar 21 11:48 ntuple_AODtoDAOD.pmon.gz
-rw-r--r--.  1 root root  13M Mar 21 11:48 log.AODtoDAOD
-rw-r--r--.  1 root root  320 Mar 21 11:48 mem.summary.AODtoDAOD.json
-rw-r--r--.  1 root root 5.9K Mar 21 11:48 jobReport.json
```

Measurement

- There is a myriad of ways to obtain system level metrics.
- For a complete list and a comprehensive review see “System Performance” by Brendan Gregg.
- It’s difficult to ask sysadmins to become experts in a range performance tools.
- For some metrics a “black box” approach may not even be viable and instrumentation may need to be added to payloads.
- The end goal is to identify:
  - A core set of information sources (such as /proc).
  - A core set of tools that parse, collect and aggregate these information sources.

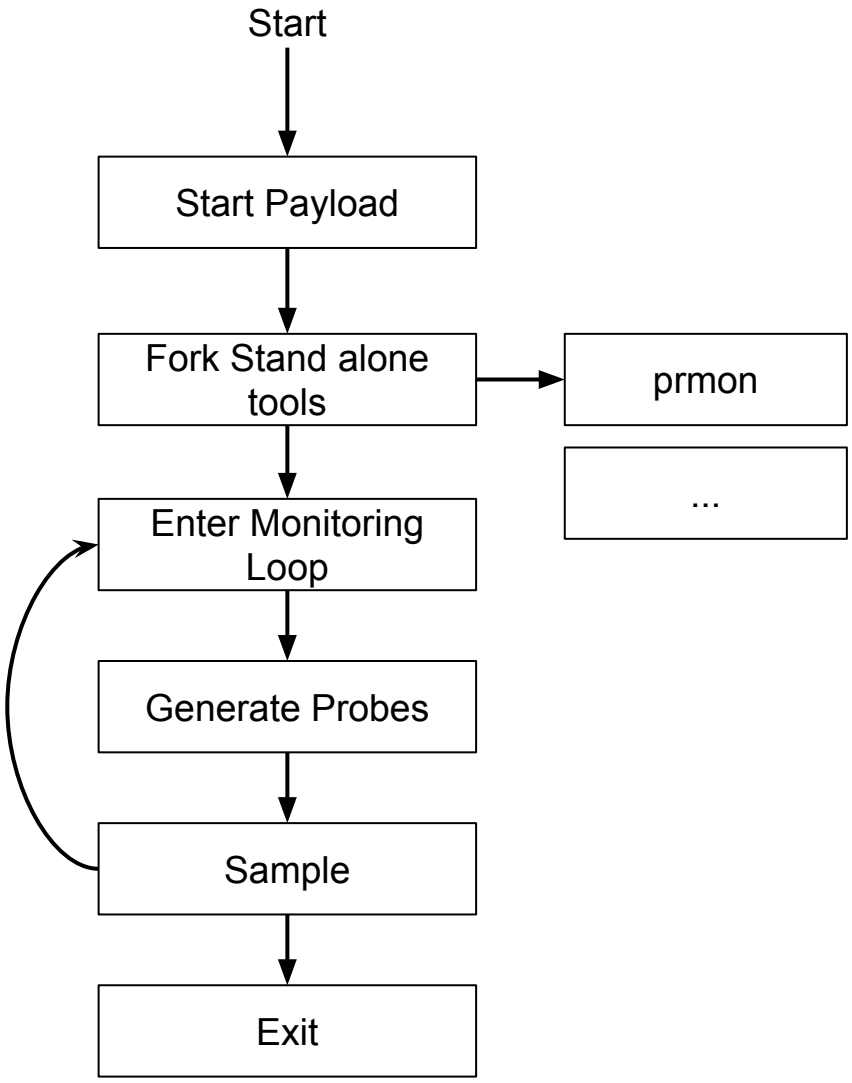


http://www.brendangregg.com/linuxperf.html



Measurement

- To make things as simple as possible it is envisaged that a single tool/framework will be built.
- This will:
  - Act as an init style process for already existing tools such as **prmon** (a.k.a. MemoryMonitor).
  - Identify all child processes of the current payload under investigation.
  - Directly or via system tools (such as **perf**) capture low level performance metrics.
- It is important to leverage tools already used in production and at scale to avoid duplication of effort.



Process Monitor (prmon)

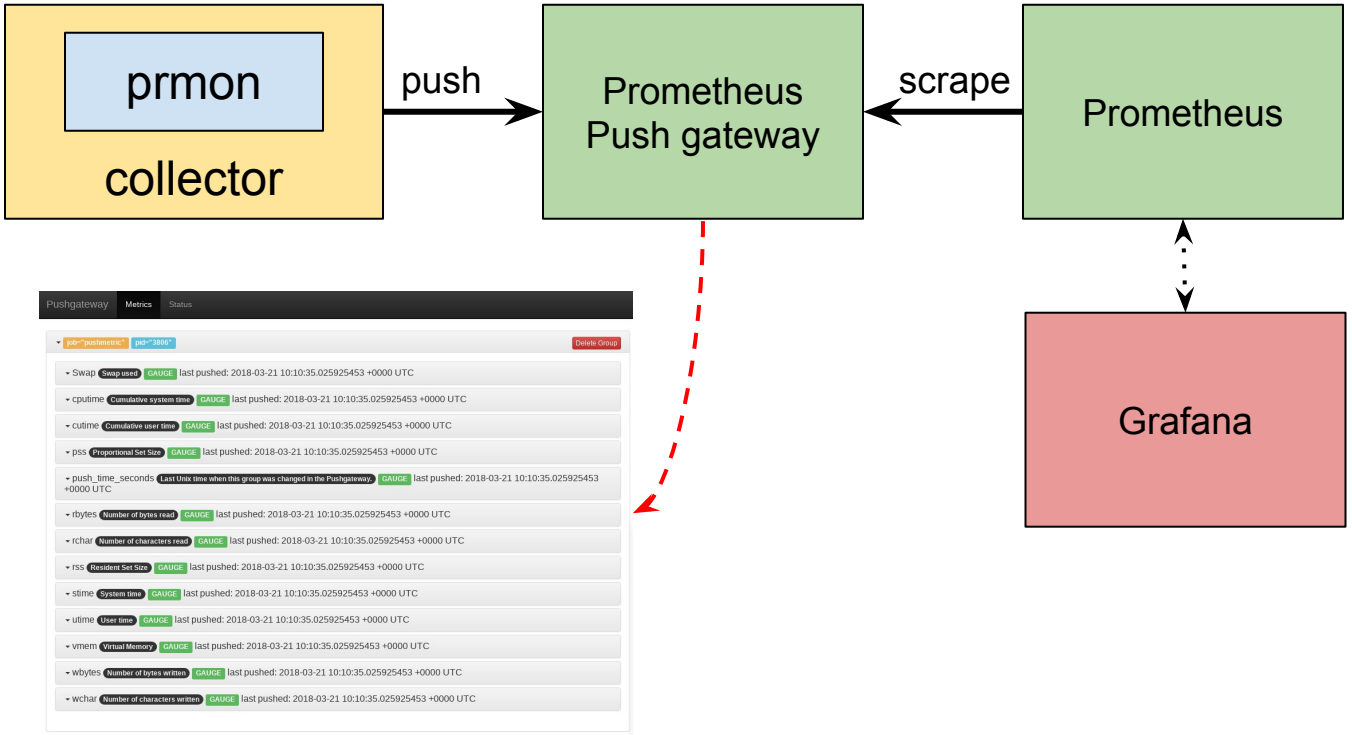
What is prmon?

- A “fork” of the ATLAS MemoryMonitor codebase used by ATLAS to collect memory and I/O information from production transformations.
- prmon read information from the **/proc** filesystem and aggregates extracted metrics across all child processes (important for looking at resource utilisation of multi-process workloads).
- prmon currently collects:
  - VMEM, PSS, RSS, Swap memory utilisation metrics
  - rchar, wchar, rbytes, wbytes I/O utilisation metrics
  - utime, stime, cutime, cstime CPU utilisation metrics
- Source available from: https://github.com/HSF/prmon

Time	VMEM	PSS	RSS	Swap	rchar	wchar	rbytes	wbytes	utime	stime	cutime	cstime
1521105584	276096	4708	9192	0	942142349	125908662	41761280	140156928	0.06	0.32	12.95	3.14
1521105586	276108	4748	9252	0	942430266	125909449	41761280	140161024	0.06	0.33	12.96	3.19
1521105588	276120	4768	9272	0	942718188	125910233	41761280	140165120	0.07	0.33	12.97	3.24

A Toy Metric Collection System

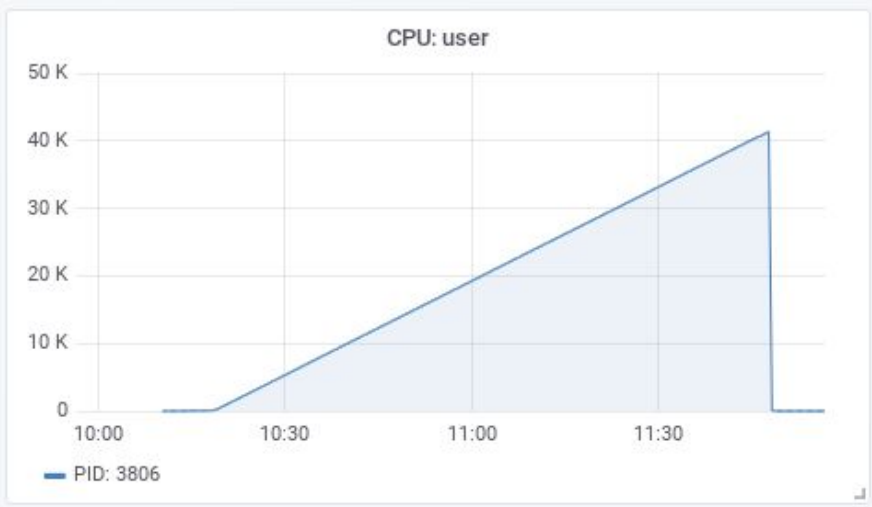
- Initially created a “toy” metric collection system to explore real payloads, and test the idea of a monitoring tool/framework.
- Using a modified version of prmon to act in a monitoring loop as a sample point and exporting metrics to a Prometheus via a push gateway.
- Visualisation & Graphing handled by Grafana.
- Initial tests with an ATLAS MC workload and an ATLAS AOD to DAOD derivation.
- Graphs and metrics shown are from the derivation run.

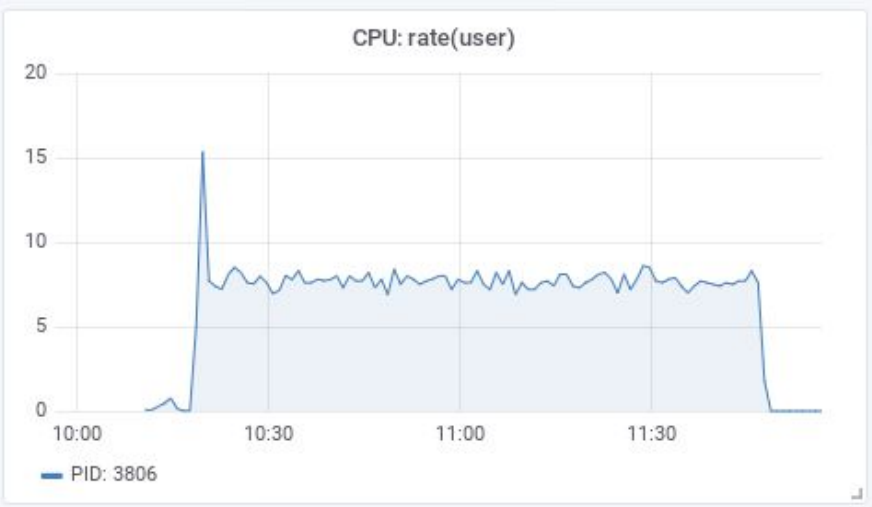


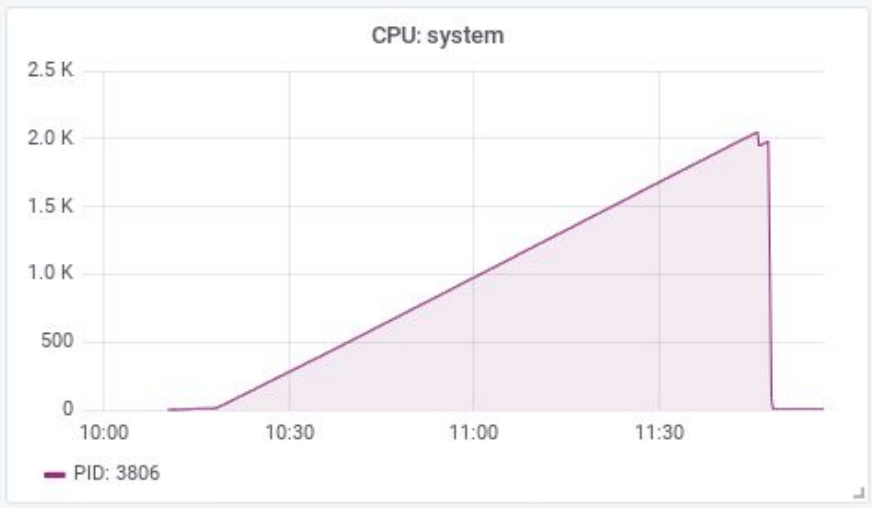
A Toy Metric Collection System

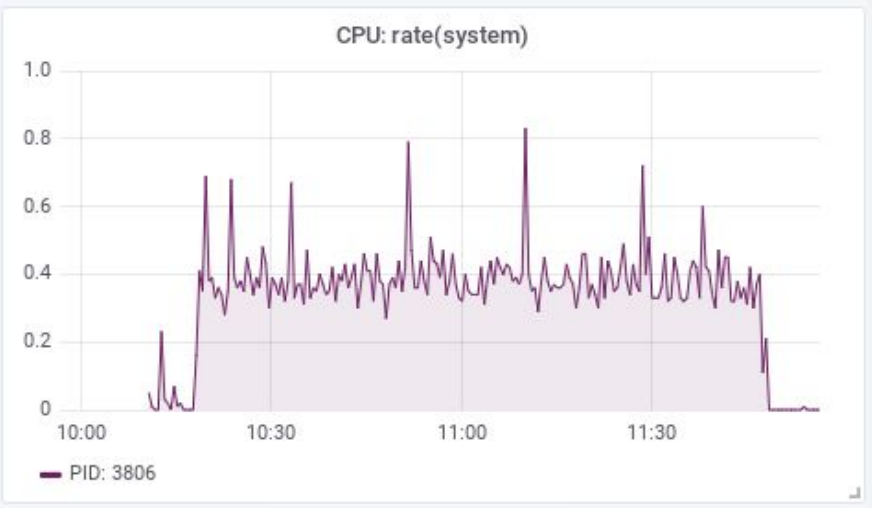
	VMEM	RSS	PSS
Average	26.1GB	14.8GB	6.6GB

A Toy Metric Collection System









A Toy Metric Collection System

Conclusions

- Detailed performance metrics are needed to assess the impact of changing WLCG payloads.
- A simple set of metrics has been identified to begin exploring payload performance.
- A tool has been designed that can form the core of a comprehensive monitoring system.
- A “toy” monitoring system has been created to help identify metrics and explore ideas presented here.