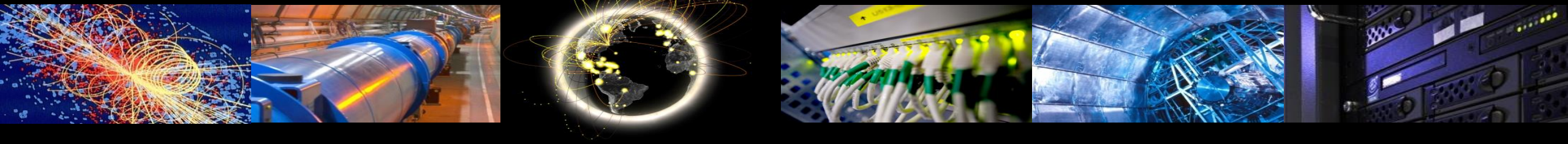
Feedback from the WLCG/HSF System Performance and Cost Modeling WG

Andrea Sciabà
on behalf of the WG

12 June 2018, pre-GDB



WLCG
Worldwide LHC Computing Grid

CERN

Outline

- Some background information on the WG
- I will discuss only the WG activities of interest to the benchmarking WG
- Some preliminary results for application metrics on reference workloads

The working group

- In a nutshell
  - Address the various aspects of the process that starts from the data processing and analysis needs of the LHC experiments to determine their needs in terms of computing and storage resources (and their cost)
  - Contribute to the reduction of the gap between our best estimates of the required and available future computing resources
- More information
  - https://twiki.cern.ch/twiki/bin/view/LCG/WLCGSystemsPerformanceModeling
  - https://indico.cern.ch/category/9733/
  - Recent HEPIX talk by Pepe (slides)

Main tasks

- Maintain a glossary of terms
- Collect reference workloads for each experiment
- Package the reference workloads with containers
- Define the best properties to characterise a workload
- Draft a cost evaluation process
- Design a toy model of a workload
- Compile a list of relevant performance analysis tools
- Set up a distributed testbed to run tests
- Create a simple resource calculation model
- Create models for the workloads at the HL-LHC scale

Reference workloads

- Each experiment provided one or more typical MC jobs to be used as “test subjects”
  - Usually derived from recent production campaigns
  - Covering the full MC processing chain
  - All documented and tested
- This allows to apply software profiling and system performance tools on a fixed target
  - Updated only when needed (e.g. to use more performant, but still production software versions)
  - NOT intended to test future improvements
  - Worthy candidates for a potential HEP benchmark?
    - Some of them require input files (to be packaged with benchmark) and all of them access CVMFS
- Workloads classified as CPU intensive or I/O intensive or both

ALICE and ATLAS

- ALICE
  - pp collision generation, simulation and reconstruction
  - Tarball available (1 event by default)
- ATLAS (docs)
  1. Geant4 simulation
     1. Input: local EVNT file from event generation
  2. Digitisation and reconstruction
     1. Sub-steps: HITtoRDO, RDOtoRDOTrigger, RAWtoESD+ESTtoAOD, merge
     2. Input: local HITS files from MC simulation + local pile-up HITS
  3. Derivation
     1. AODtoDAOD (multiple DAOD)
     2. Input: local AOD
  - Now optionally also remote access from EOS

CMS and LHCb

- CMS (docs)
  1. GENeration-SIMulation (ttbar events)
  2. Digitisation, trigger and pileup
     1. Input: remote access via xrootd
  3. Reconstruction and analysis data creation
     1. Input: remote access via xrootd
- LHCb (docs)
  1. Gauss: generation and G4 simulation
  2. Boole: digitisation
  3. Moore: trigger emulation
  4. Brunel: offline reconstruction
  5. DaVinci: stripping

  – All steps chained together in the same job

Properties to characterise workloads

- Two-fold task
  - Choose which properties we consider most useful to describe a workload
  - Have tools to automate the collection and analysis (e.g. plotting) of these properties (metrics)
  - Iterative process: choose metrics, measure them, assess their usefulness, add more metrics

Metric lists: CPU and memory

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

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)

From HSF/WLCG workshop 2018

Metric lists: IO and network

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	

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

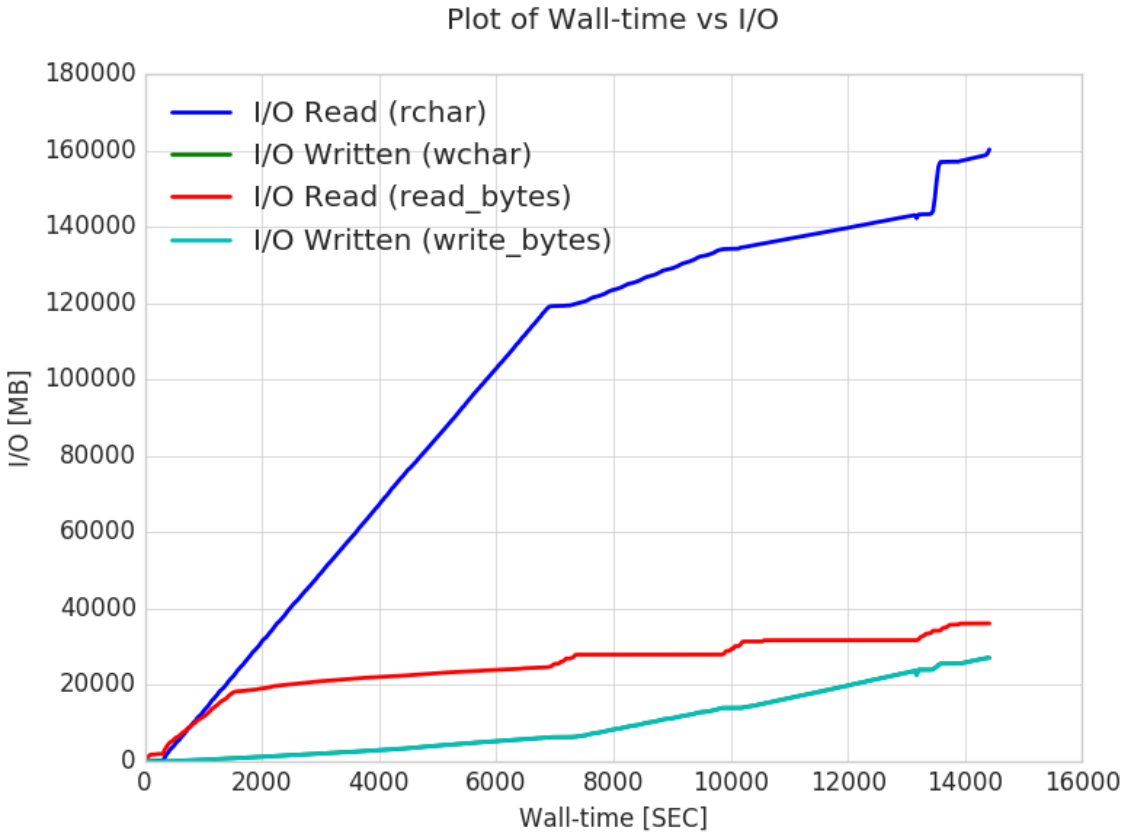
From HSF/WLCG workshop 2018

prmon

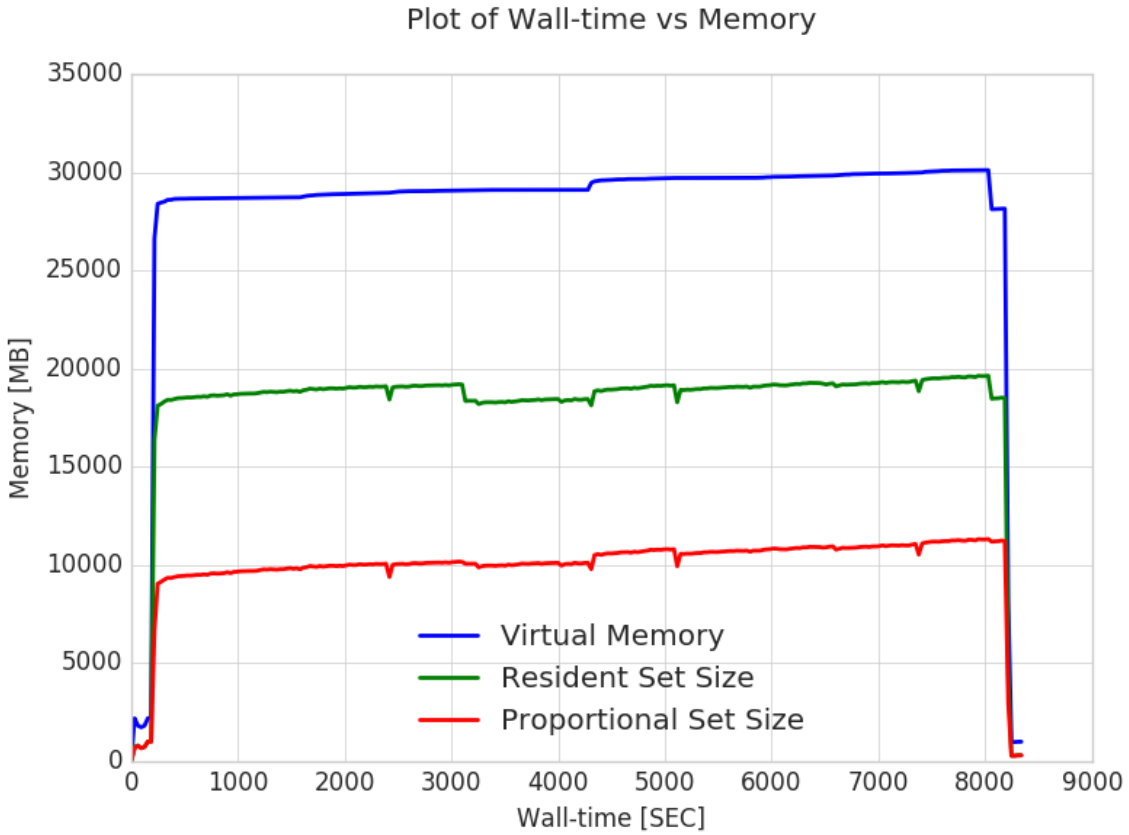
- Tool to monitor resource consumption of a process and its children (repository)
  - Derived from ATLAS' MemoryMonitor but made completely application-agnostic
  - Main developers: Graeme S. and Serhan M.
  - Version 1.0.0 released, soon will add binaries for download
- Metrics included cover most of what is in the previous tables
  - VMEM, RSS, PSS
  - rchar/wchar (bytes read/written by the process) , read_bytes/write_bytes (bytes read/written from/to the storage layer)
  - User time, system time, wallclock time
  - xx_bytes, tx_bytes, rx_packets, tx_packets
- Already applied to most of the reference workloads
  - Using 8 processes/threads when applicable
  - (CMS RECO and LHCb still to be done…)

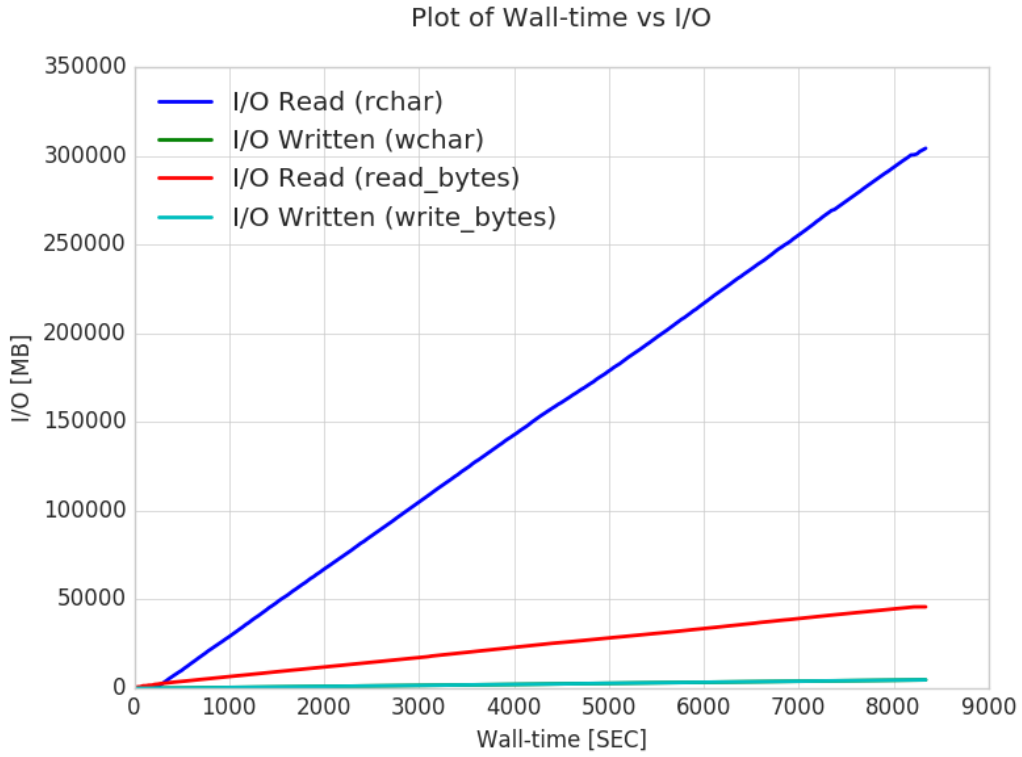
ATLAS simulation

ATLAS Digi+Reco

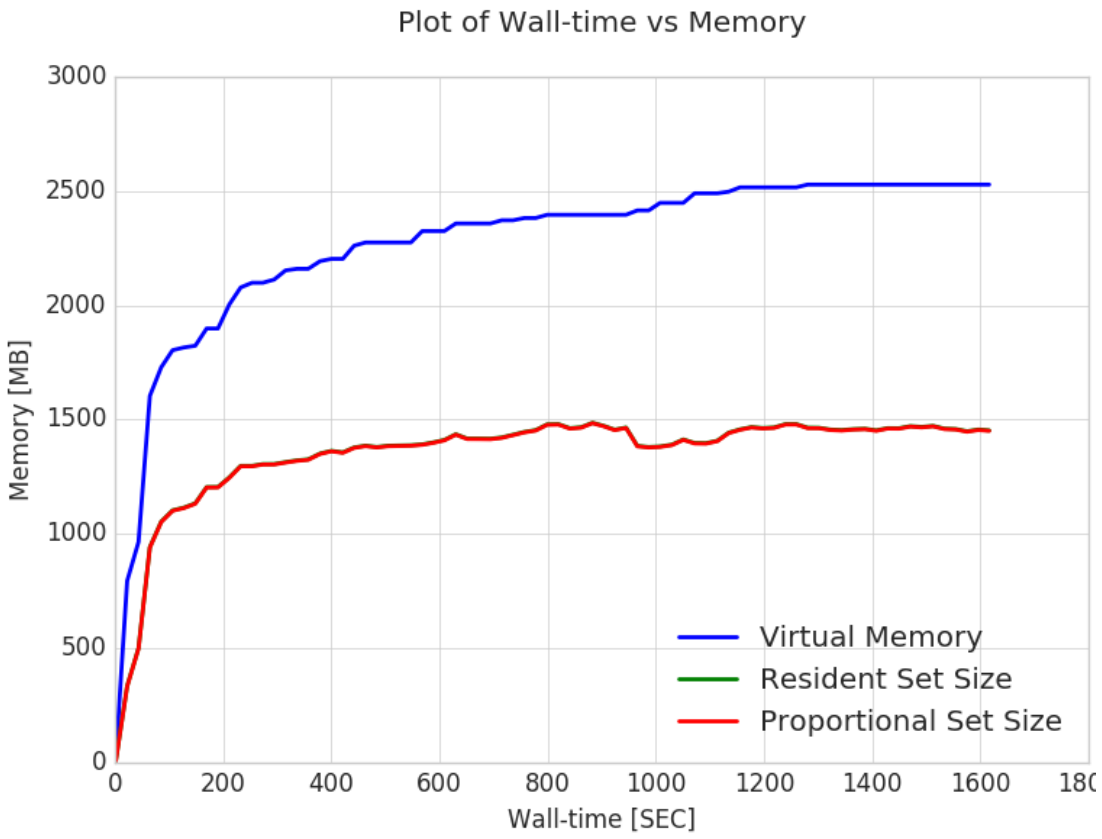


ATLAS derivation



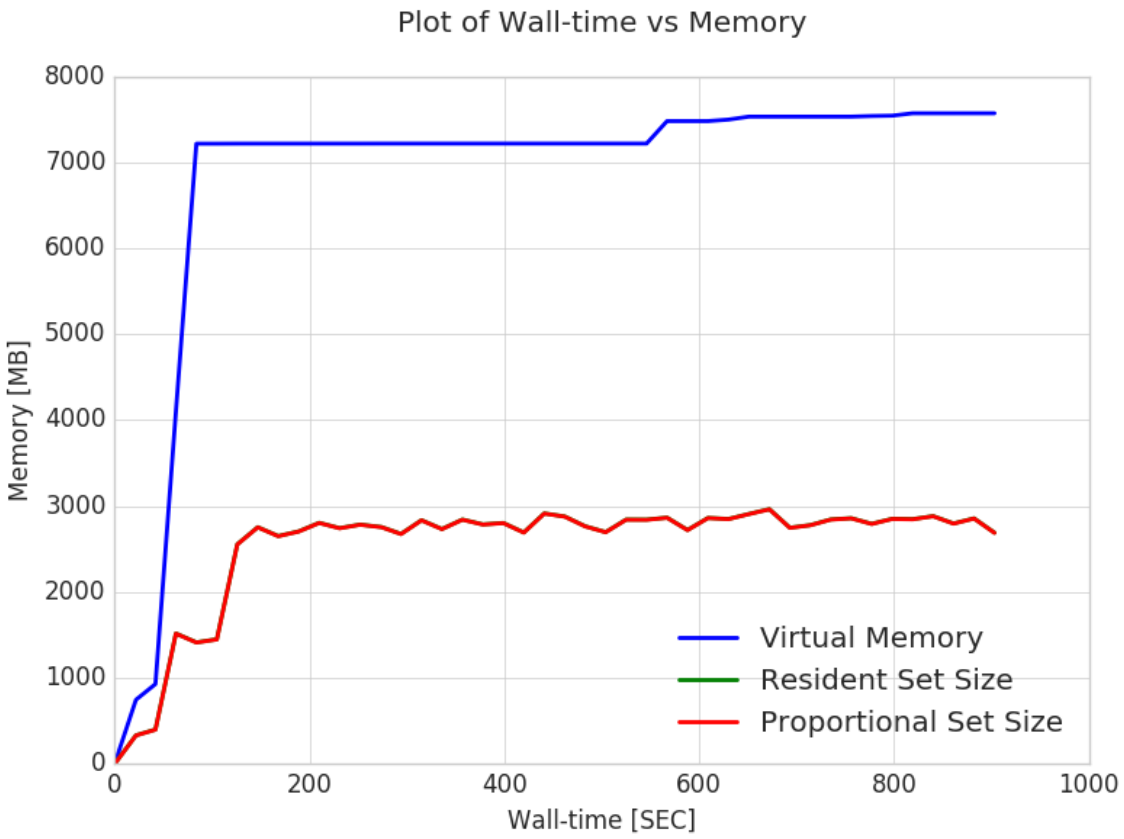


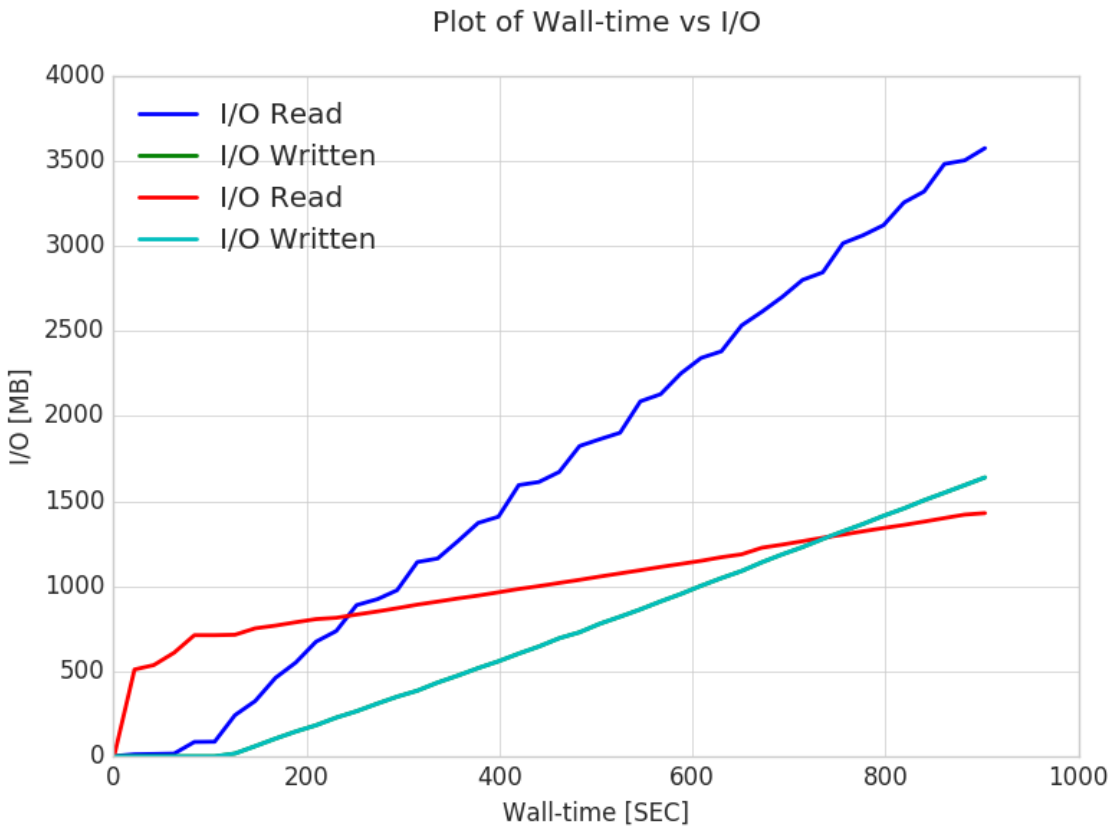
CMS GENSIM


Plot of Wall-time vs Memory
Memory [MB]
Wall-time [SEC]
Virtual Memory
Resident Set Size
Proportional Set Size

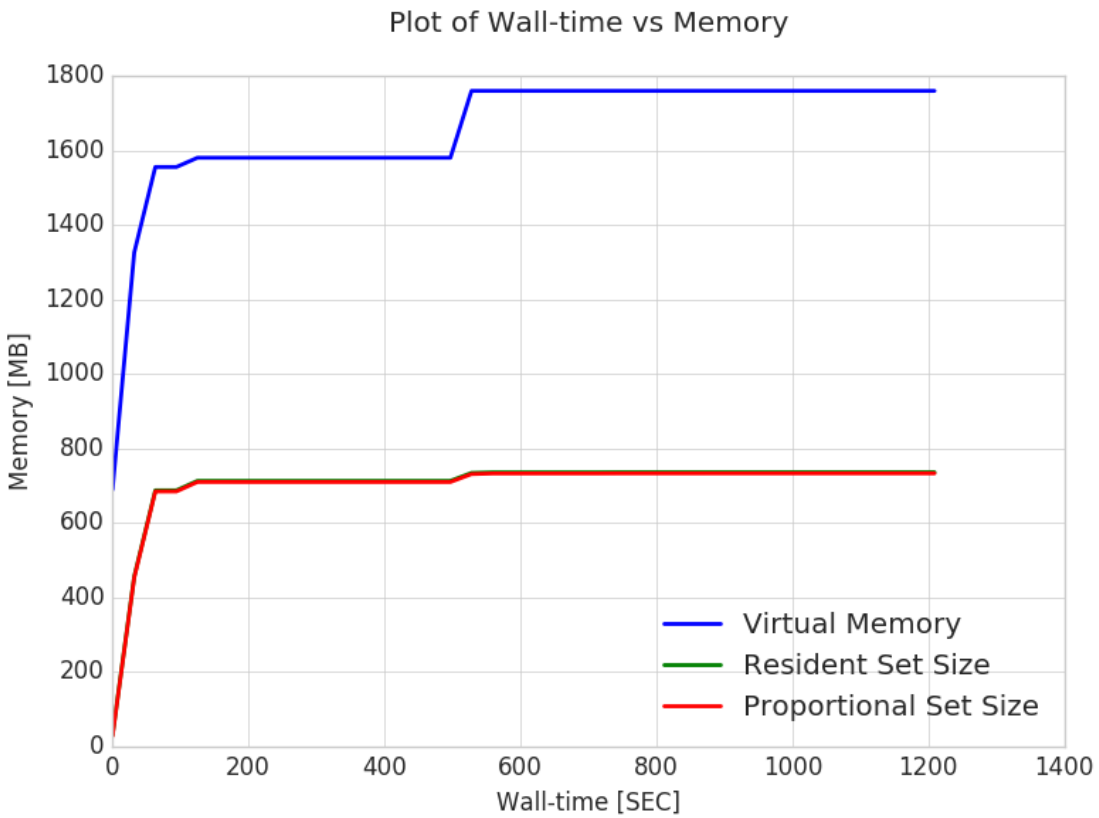

Plot of Wall-time vs I/O
I/O [MB]
Wall-time [SEC]
I/O Read
I/O Written
I/O Read
I/O Written

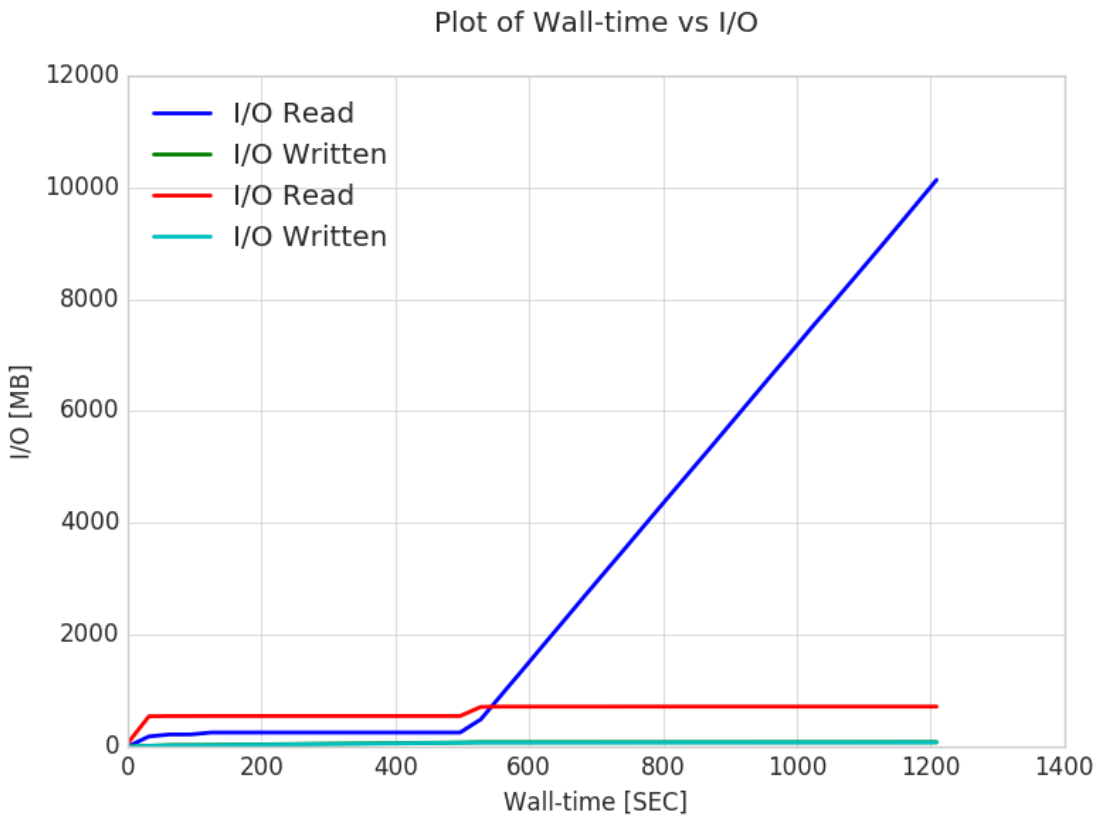
CMS DIGI


Plot of Wall-time vs Memory
Memory [MB]
Wall-time [SEC]
Virtual Memory
Resident Set Size
Proportional Set Size
0
1000
2000
3000
4000
5000
6000
7000
8000
0
200
400
600
800
1000


Plot of Wall-time vs I/O
I/O [MB]
Wall-time [SEC]
I/O Read
I/O Written
I/O Read
I/O Written
0
500
1000
1500
2000
2500
3000
3500
4000
0
200
400
600
800
1000

ALICE pp


Plot of Wall-time vs Memory
Memory [MB]
Wall-time [SEC]
Virtual Memory
Resident Set Size
Proportional Set Size
0
200
400
600
800
1000
1200
1400
1600
1800


Plot of Wall-time vs I/O
I/O [MB]
Wall-time [SEC]
I/O Read
I/O Written
I/O Read
I/O Written
0
2000
4000
6000
8000
10000
12000
200
400
600
800
1000
1200
1400

Metric summary

Type	Events	Duration (hours)	CPU efficiency (%)	PSS/process or thread (MB)	Disk read rate (kB/s)	Disk write rate (kB/s)	Network traffic (kB/s)
ATLAS sim	1000	9.4	98	500	140	70	negligible
ATLAS digi reco	2000	4.0	84	1500	2600	1900	negligible
ATLAS derivation	?	2.3	96	1400	5600	580	negligible
CMS GENSIM	500	0.5	97	200	600	240	negligible
CMS DIGI premix	500	0.25	58	400	1600	1900	3300
ALICE pp	1	0.3	100	700	600	60	negligible

Other performance analysis tools

- Trident (see next talk by Servesh)
- Comprehensive review from at Naples workshop (slides)

Tool	CPU	Memory	IO	System	App	Overhead	Annotate	Need recompile	
Linux perf	✓	✓	✓	✓	✓		✓	✗	
gperftools	✓	✓	✗	✗	✓		✓	✗	
gprof	✓	✗	✗	✗	✓		✓	✓	
valgrind	✓	✓	✗	✗	✓		✓	✗	
iostat	✓	✗	✓	✓	✗		✗	✗	
vmstat	✓	✓	✓	✓	✗		✗	✗	
/proc	✓	✓	✓	✓	✓		✗	✗	
ICC Report	✓	✓	✗	✗	✓		✓	✓	
VTune	✓	✓	✗	✗	✓		✓	✗	
Advisor	✓	✓	✗	✗	✓		✓	✗	
prmon	✓	✓	✗	✗	✓		✗	✗	
IgProf	✓	✓	✗	✗	✓		✓	✗	
MALT	✗	✓	✗	✗	✓		✓	✗	
NUMAPROF	✗	✓	✗	✗	✓		✓	✗	
FOM	✗	✓	✗	✗	✓		✓	✗	
Trident	✓	✓	✓	✓	✗		✗	✗	

Conclusions

- Several points of interaction between the two working groups
- Application characterisation should be done coherently
  - To be applied also to prospective HEP benchmarks
- Prmon proposed as the place where to centralise the collection of systems performance metrics
  - In particular for new development, it can coexist with other tools