WMS status

JRA1 All Hands Meeting
Catania, 7-9 March 2007

Marco Cecchi – INFN / CNAF

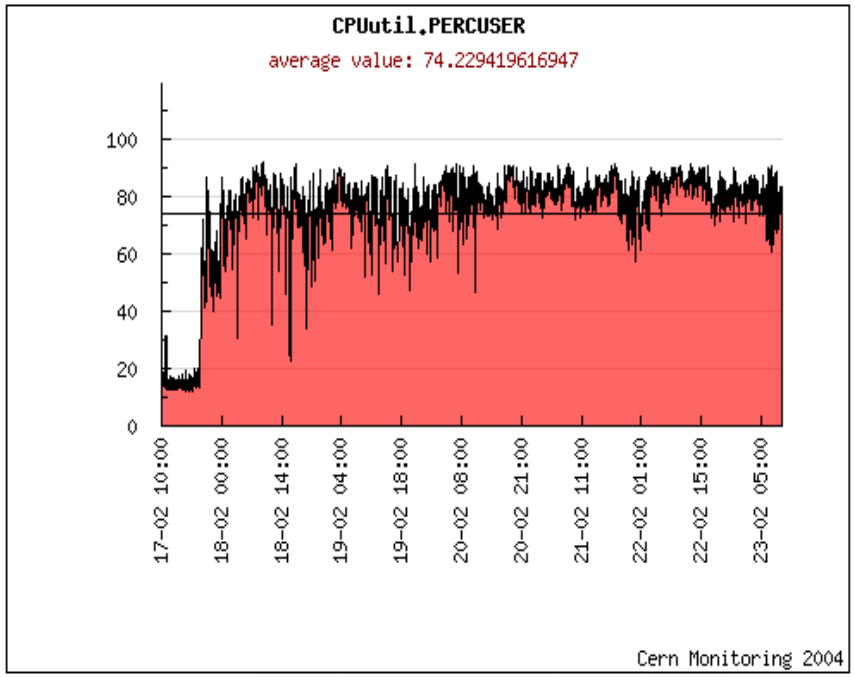
Outline

- Status of the recent WMS 31 tests @ CERN
- Memory management issues (wmp, wm, ctpl)
- Design issues (MM, ISM)
- Bulk Match-Making tests
- WMS requirements and acceptance criteria

Testing WMS 3.1

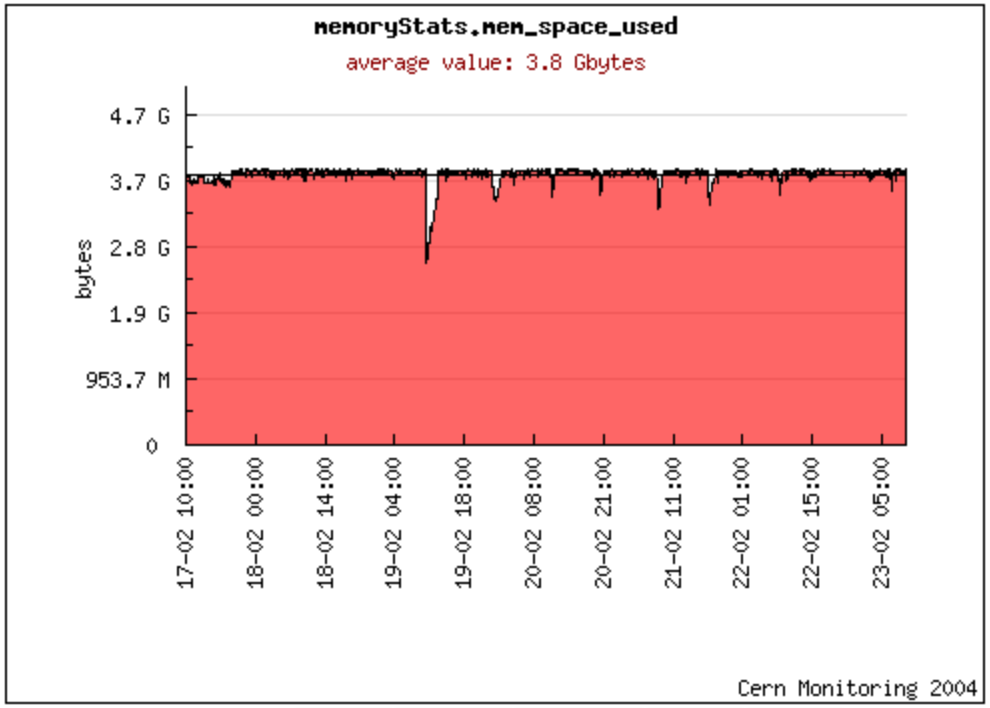
- **lxb7283@CERN (2 CPUs: Intel Xeon 2.80GHz / 4 Gb RAM)**

CPU:



	Day	Month	Year	Hour	Minute
Start time:	17	Feb	2007	10	38
End time:	23	Feb	2007	10	17

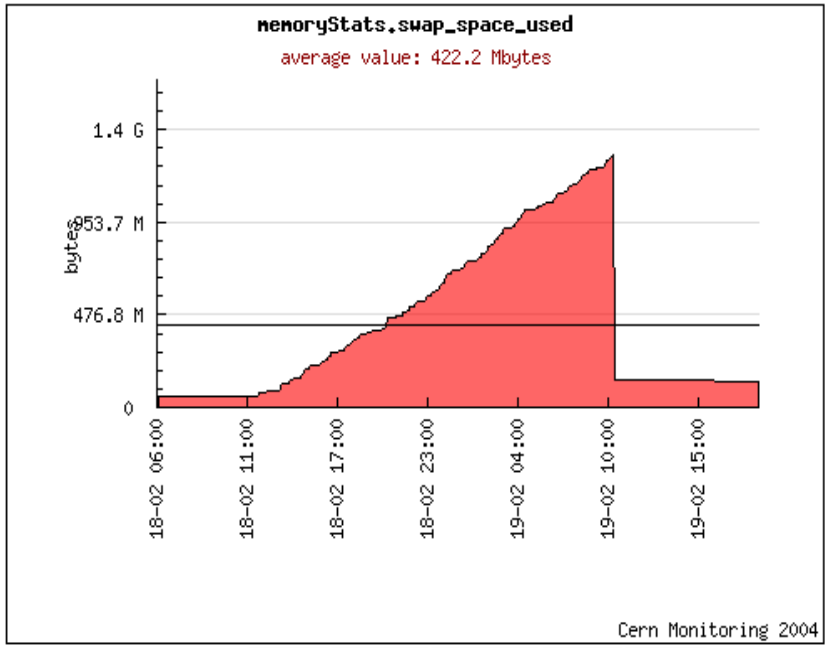
Memory:



	Day	Month	Year	Hour	Minute
Start time:	17	Feb	2007	10	38
End time:	23	Feb	2007	10	17

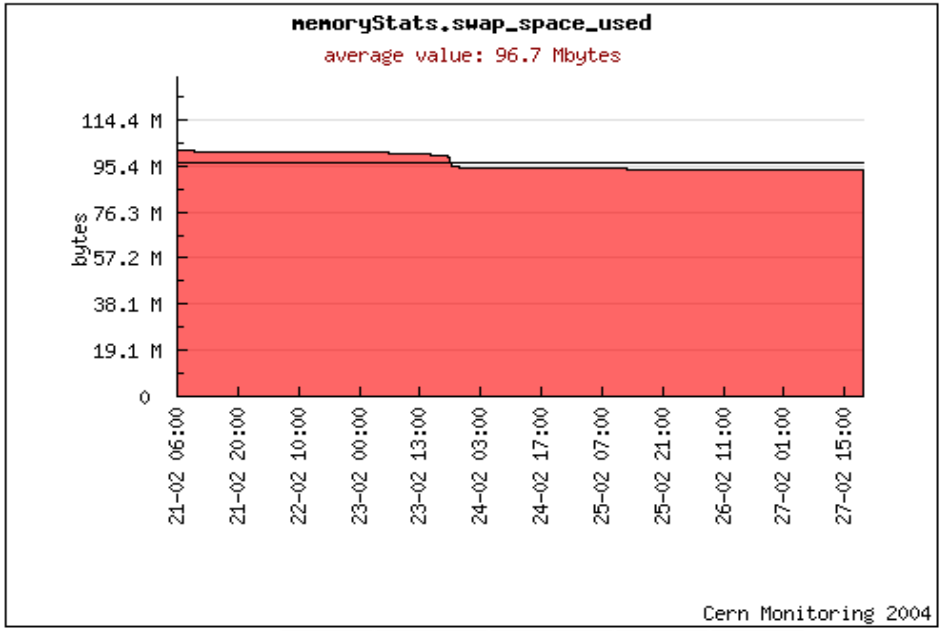
Testing WMS 3.1

Swap usage before the suicidal patch:



	Day	Month	Year	Hour	Minute
Start time:	18	Feb	2007	6	26
End time:	19	Feb	2007	19	26

After the cure:



	Day	Month	Year	Hour	Minute
Start time:	21	Feb	2007	6	26
End time:	27	Feb	2007	19	26

Google malloc

Google performance tools

http://code.google.com/p/google-perftools/

- **What we usually refer to as 'google malloc'**
- **They claim this is the fastest malloc ever**
- **It works well with threads and STL (what we can witness)**
  - We have a std::map accessed by several threads
  - Per-thread arenas (std) hold back released memory for later use. Memory blowups that we could have never spot with a profiler follow.
- **Perf Tools is distributed under the terms of the BSD License.**
- **Source code tarball available**
- **libtcmalloc_minimal.so = 50k (we don't need profiling capabs)**

MM performance I

- **WM_Logfile@lxb7283 from Feb, 23rd 10AM to Mar 3rd 8PM**

[root@lxb7283 glite]# grep "MM for job:" workload_manager_events.log1|wc -l

2745

```
...
https://lxb7026.cern.ch:9000/KxQsQq19CnXIkOktSqppAg 3.91
https://lxb7026.cern.ch:9000/roQspp0Guk8bUbI_hr6iQQ 2.58
https://lxb7026.cern.ch:9000/u-LDlwHG-N7hzE3NjARknA 2.56
https://lxb7026.cern.ch:9000/VP_MhFOfkLYCkVgDv6U0eg 3.64
https://lxb7026.cern.ch:9000/u-LDlwHG-N7hzE3NjARknA 2.57
https://lxb7026.cern.ch:9000/jnRt_mG-Gurr2vO7AUJoCw 2.59
https://lxb7026.cern.ch:9000/u-LDlwHG-N7hzE3NjARknA 2.57
https://lxb7026.cern.ch:9000/d-2UV-XekWDgLx1dmaCaOQ 2.16
https://lxb7026.cern.ch:9000/SGHhxu6RWqAEc_WrxVsrlw 3.42
https://lxb7026.cern.ch:9000/KxQsQq19CnXIkOktSqppAg 2.56
https://lxb7026.cern.ch:9000/Lgk8KmviXsAFiByFlTocsw 2.55
https://lxb7026.cern.ch:9000/6AQyoPnxS_WPfc-hPjLxrA 4.29
https://lxb7026.cern.ch:9000/ljMGOxsrBzz6M-dnoqs6yQ 2.55
https://lxb7026.cern.ch:9000/ljMGOxsrBzz6M-dnoqs6yQ 2.54
https://lxb7026.cern.ch:9000/-Xaa-fP2K0x1FKChrh7DAg 3.85
https://lxb7026.cern.ch:9000/-Xaa-fP2K0x1FKChrh7DAg 2.55
https://lxb7026.cern.ch:9000/1NJh2cf3hejX5ziqJsJmOg 3.93
https://lxb7026.cern.ch:9000/HKpwgu8jgywTll454GUqww 2.58
https://lxb7026.cern.ch:9000/ogyyibw-k7oWQPB6RsXLaw 3.43
https://lxb7026.cern.ch:9000/xzSbUEZVLb_nMmAWgvv3Lg 2.57
....
```

=AVERAGE(A1:A2745)

2.82 secs (estimated to be ~4secs before the latest optimizations)

Even if using collections the impact of the MM performance is reduced by a factor 1 to #clusters (~[10-100]) we are still asked to keep on looking after the MM performance because submission of single jobs or jobs in a high number of clusters might always occur.

MM performance II

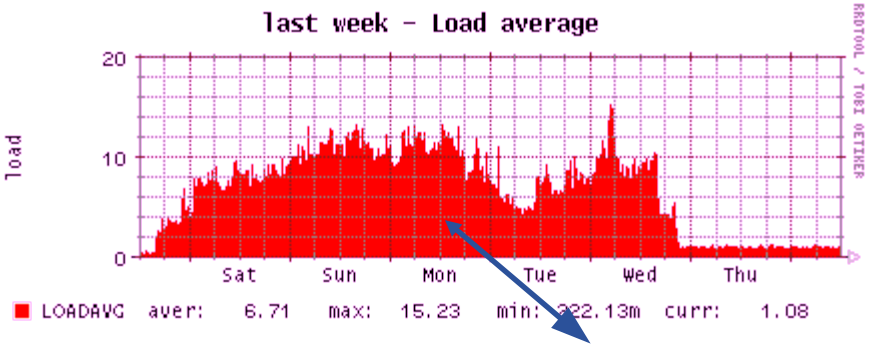
- **Margins are still at reach if we manage to do it in //**

- **Now it is done sequentially**
  - **ISM concurrent r/w access by multiple threads**
  - **intrinsic classad library limitations**
  - **yet ~3sec/MM is 28.8kjobs/day**
  - **just recall that DAGless means 28k*100 by the WM (if the others processes keep up)**

Bulk Match-Making tests

Bulk-Match-Making (AKA DAGless) tests

- 5 days uninterrupted submission
- 15 Kjobs/day rate
- *hello world* jobs with real ATLAS req's
- Proxies expiring after 5 days (on purpose)



Load under control

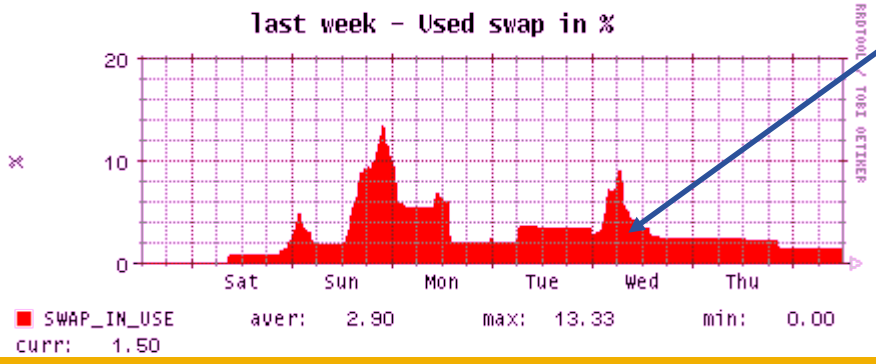
Site	Submit	Wait	Ready	Sched	Run	Don(S)	Don(F)	Abort
ce05-lcg.cr.cnaf.infn.it	0	19	0	0	0	5345	0	536
cclcgceli02.in2p3.fr	0	22	0	0	0	5307	1	470
ce04.pic.es	0	32	0	0	0	5264	10	494
ce-fzk.gridka.de	0	20	0	0	3	5285	5	487
lcg00125.grid.sinica.edu.tw	0	28	0	0	0	5268	4	500
lcgce01.gridpp.rl.ac.uk	0	24	0	0	0	5380	4	492
lcgce01.triumf.ca	0	17	0	0	0	5217	10	556
ce101.cern.ch	0	32	0	0	0	5158	4	606
ce102.cern.ch	0	29	0	0	0	5082	3	686
ce108.cern.ch	0	18	0	0	1	5126	1	654

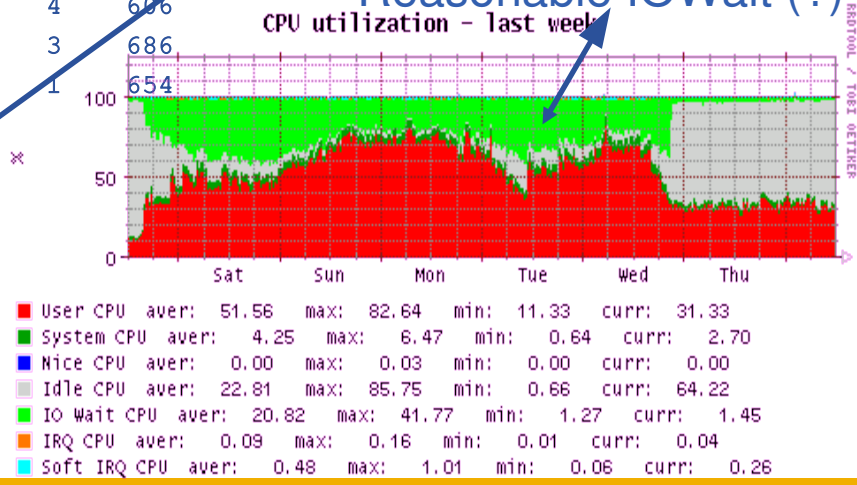
Aborts mostly due to proxy expiration

< 0.5% jobs in non-final states

Memory usage under control

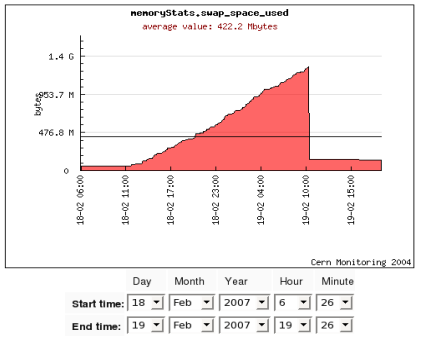
Reasonable IOWait (?)





Wmproxy suicidal patch

- **Linear growth of memory usage**
  ⇒ memory leaks



- **Short-term solution: self-termination of the WMP server processes on a given threshold of serving requests.**
  - No service interruption
- **FCGI does't seem to be surprised so it's the workaround works fine**
- **The memory leaks (mostly in JDL and gSOAP layers) are under investigation for a long term solution.**

DAG counted planner

- Each DAG is managed by a condor_dagman process
- For each node, the planning is done invoking an external program (planner)
  - The planner is memory-hungry
- It's possible to limit the number of concurrently running planners per DAG
- But there is no “condor” way to limit the total number of planners
- Now the real planner is invoked through a proxy executable that first checks the total number of already running planners through the /proc dir

Requirements and Acceptance Criteria

Ian Bird, 1st March, 2007

…"A single WMS machine should demonstrate submission rates of at least 10K jobs/day sustained over 5 days",...

- **That's easily at reach. Also, already proved to be sustainable.**

…"This means that issues of memory consumption and growth, file systems filling, etc. must be resolved"…

- **(almost?) Done.**

ATLAS has stability needs to be able to get along with the WMS.

- **It's much better now, especially DAGLESS (0.5%) vs DAGMan (5%) collections is proving to be faster and more reliable.**
- **Modifications are needed and are being applied also to the WMProxy and the User Interface.**