Performance testing of L&B service

M. Voců, A. Křenek, J. Pospíšil, M. Mulač, F. Dvořák
CESNET, Czech Republic

www.eu-egee.org

Information Society

Overview

We will talk about ...

- motivation (why?)
- methodology (what?)
- implementation (how?)
- results (omg!)
- conclusion & future work (what should we do?)

Motivation

We want to . . .

- decrease latencies when submitting (large) jobs
- increase performance of **event logging** and **job registration**
- identify bottlenecks
- determine impact of code changes on performance
- measure performance using real jobs in real environment

. . . automatically

Motivation

We want to . . .

- decrease latencies when submitting (large) jobs
- increase performance of **event logging** and **job registration**
- identify bottlenecks
- determine impact of code changes on performance
- measure performance using real jobs in real environment

. . . automatically

Final goal:

- achieve throughput of 10^6 **jobs/day** (megajob/day)

Methodology

How is job modeled?

- typical series of events
- sample jobs chosen from **live** L&B database (tigerman)
- four categories:

average job

- 10 events, average size 0.5 KB
- DAG node, no problems

big job

- 150 events, average size 360 B
- DAG node, 6 resubmissions, unable to find match, X509 proxy expired

average DAG

- 13 events, average size 0.8 KB
- DAG, 3 subjobs, no problems

big DAG

- 31 events, average size 110 KB (but 2 events with 1.5 MB size)
- DAG, 1000 nodes, subjob cancel caused DAG abort

Methodology

Standard event logging:

- measure real time required for given number of jobs (i.e. their events) to travel through L&B component(s)
- parts of the component, one component at a time or the whole component chain
- **asynchronous** nature of L&B $\Longrightarrow$ we have to measure at both ends
- events are sent by special producer, prepared beforehand
- events are consumed by calling special library function inside component or by the next component

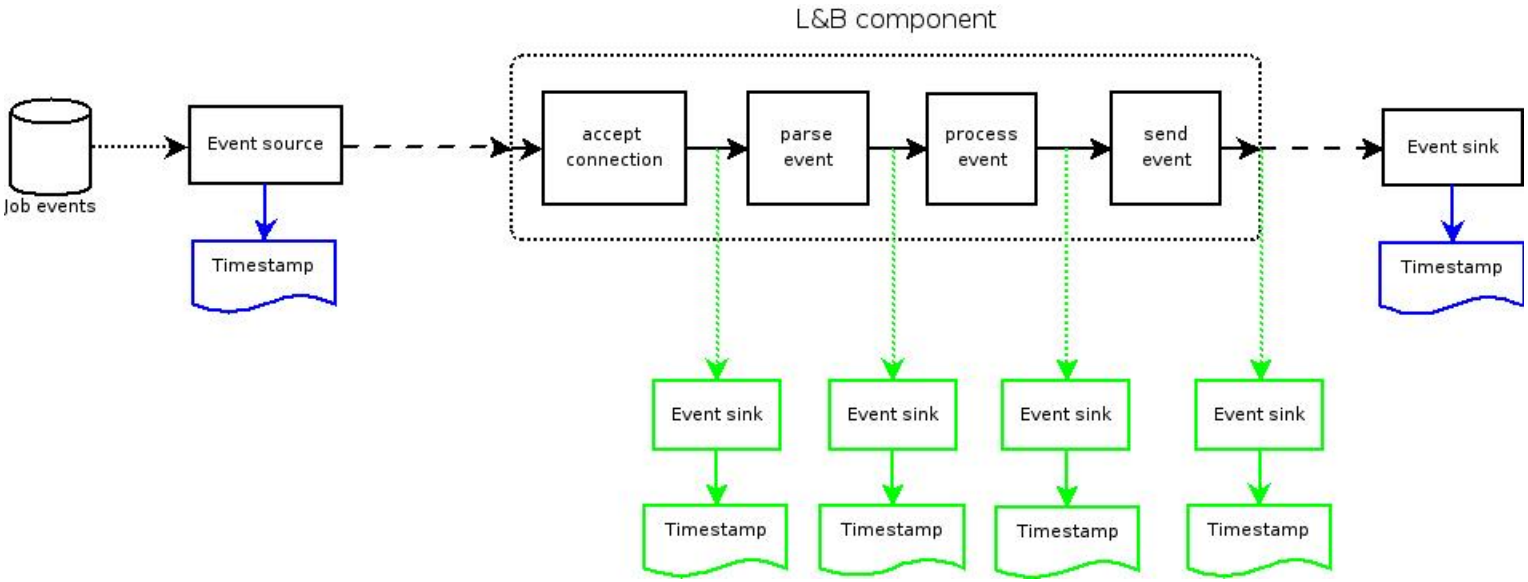
Methodology

Job registrations:

- measure real time required to register DAG with given number of subjobs
- **synchronous** $\Longrightarrow$ can be measured by timing client

Methodology

Overview:



- event source logs information about used input file and timestamp of the **first event**
- event sink logs timestamp of the **last event**

Implementation

Test implementation:

- all code is in CVS, enabled by defining `LB_PERF` env. variable
- instrumented components, test scripts and input files are staged
- test script reads log files, computes test results and pretty-prints result table
- component tests include variants with different sink points

Results

How we tested:

- reference machine (scientific, Linux 2.4.21, P4 3.2GHz, 1GB RAM, UltraATA/100 WDC WD1600JB disk)

Component test:

- identifies bottlenecks within isolated component
- events consumed by the next component in chain

Chain test:

- simulates real environment
- events logged through L&B proxy & interlogger to the L&B server on another machine
- measures time required to deliver and store events at the server
- includes arbitratry load on the server (parallel queries)

Results

Example — individual test results:

```
umbar$ ./stage/sbin/perftest_proxy.sh 1000
-----------------------------------
LB Proxy test
-----------------------------------
Events are consumed:
1) before parsing
2) after parsing, before storing into database
3) after storing into db, before computing state
4) after computing state, before sending to IL
5) by IL
```

	avg_job	big_job	avg_dag	big_dag
1)	56425742	3776652	40517118	531681
2)	48265244	3267824	34150899	356861
3)	7627352	398125	3454849	0
4)	4418998	268419	3217028	0
5)	3149848	212856	2626196	0

Results

Logging results (mjobs/day):

Component	*Average job*	*Big job*	*Average DAG*	*Big DAG*
Test producer	153.6	11.2	101.5	0.6
Locallogger	0.1	0.007	0.09	0.02
Interlogger	2.5	0.2	2.7	0.1
Proxy	1.3	0.2	1.5	x
Proxy→Server	0.6	–	–	–
Proxy→Server (with queries)	0.5	–	–	–

Results

Registration results:

Component	1	1000	5000	10000	subjobs
Server	0.04	4.9	6.2	6.3	(msubjobs/day)
Proxy	1.3	4.7	7.4	8.0	(msubjobs/day)

Conclusion

Conclusion:

- we measure logging performance by automated scripts
- megajob — are we there yet?

Conclusion

Conclusion:

- we measure logging performance by automated scripts
- megajob — are we there yet?

Some operations are very expensive:

- executing programs from shell (loading many shared libraries)
- establishing SSL connections
- too many (small) database operations

Some operations are less expensive:

- disk operations (reading/writing files)
- parsing or scanning large events

Conclusion

Improvements to the L&B:

Server & proxy

- faster duplicity check
- transactional database
- faster DAG registration

Interlogger

- lazy connection close & fast reopen
- faster input

To be done:

Locallogger, clients

- connection pool

Job registration

- parallel registration to proxy & server

Future work

Future work:

- more representative selection of jobs
- measuring on full database
- make level of test detail optional
- regular automated testing (in cooperation with SA3)

. . . end

Thank you!