

WP4 - Monitoring

Progress report

Overview

- Work completed
 - New monitoring sensor agent (MSA)
- Work in progress
 - Alarm display
 - Repository : database and server
 - Transport
- Work to be done

MSA

- Release 1.3
- Deployed on ~325 nodes at CERN
- New features:
 - Smart reconfiguration
 - Multiple transport mechanism supported
 - Self-monitoring capabilities
 - Tree-like configuration
 - Works on Solaris

Alarm display

The screenshot shows the 'EDG-fabric alarm display' window. On the left is a tree view under 'Service view' with categories like 'Compute Elements', 'Hardware resources', 'AFS', and 'LSF'. A context menu is open over a 'no contact' node, showing options: Acknowledge, Show input metrics, History, Clear, Cut, Copy, and Paste. In the center, a 'No Contact' dialog box is displayed with a list of 'Correlated metrics' for various testbeds (testbed00 to testbed07) showing 'no contact' or 'ALARM_RAISED' status. On the right, an 'Alarms help' window is partially visible, and below it, an 'Alarms and events' log shows a list of events with timestamps and descriptions such as 'High load', 'Low memory', 'lim missing', 'File system full', and 'Request stuck'.

To be released in July

Repository

- Multithreaded server being implemented
- Currently, data still stored in flat files.
- Two relational databases being tested :
 - MySQL
 - Oracle

Repository : database structure

- Database aimed at storing samples:
 - Metric identifier WHAT
 - Target identifier WHERE
 - Timestamp WHEN
 - Value
- Value is very different from one metric to the other.

Repository : database structure



One table per metric

Repository : database structure

CPU load

| Timestamp | Target | User | Nice | System | Idle |
|------------|---------|------|------|--------|------|
| 1022146593 | Tbed023 | 12 | 30 | 4 | 54 |
| 1022146620 | Tbed026 | 88 | 0 | 11 | 1 |
| 1022146653 | Tbed023 | 15 | 27 | 2 | 56 |

Repository : database structure

Network traffic

| Timestamp | Target | Interface | Read | Write |
|------------|---------|-----------|------|-------|
| 1022146593 | Tbed023 | lo | 25 | 4 |
| 1022146593 | Tbed023 | eth0 | 1341 | 11 |
| 1022146620 | Tbed026 | eth1 | 42 | 168 |

Repository management

- Not clear what to use for subscription mechanism
- A simple ASCII protocol might be used
 - Server side already implemented, missing client implementation of C API and subscription.

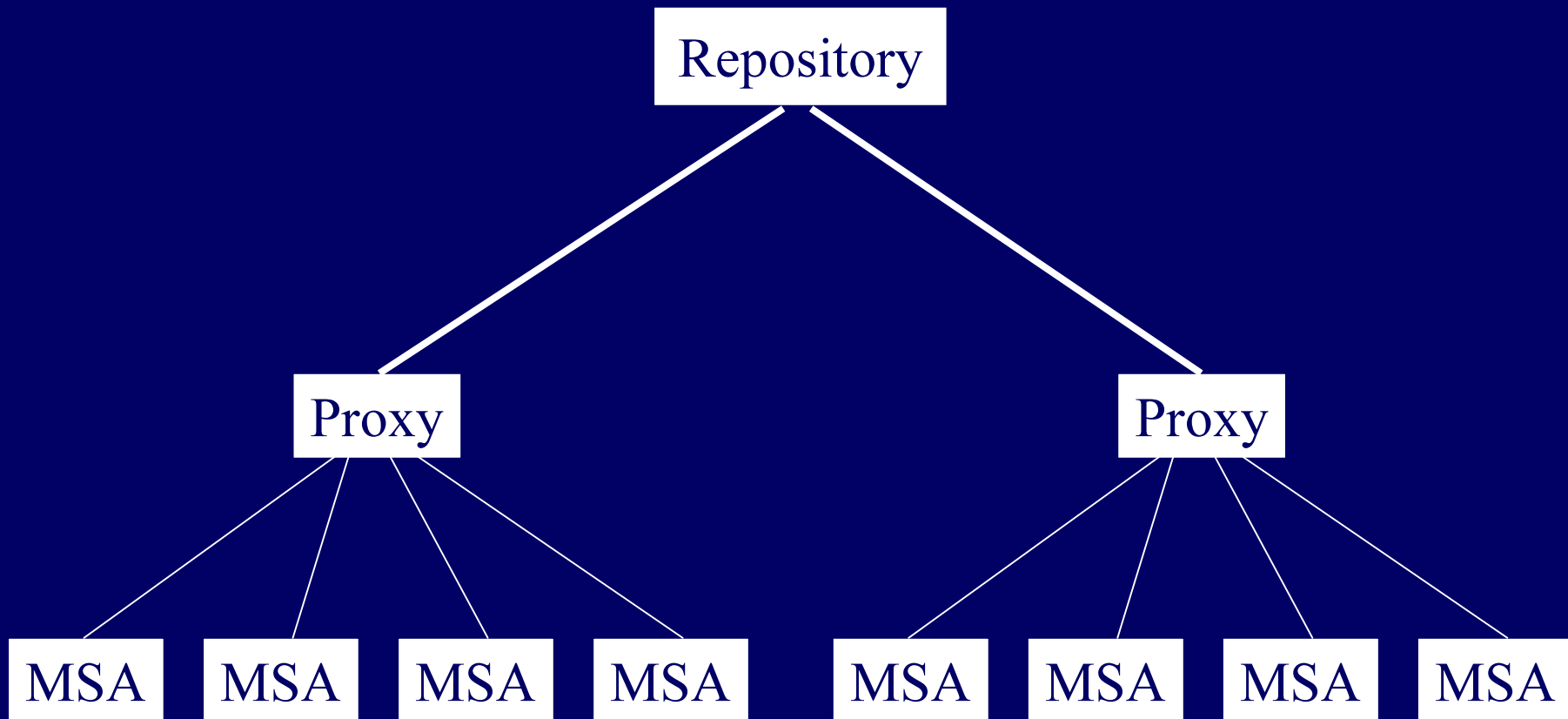
Transport

- UDP transport easy to use and deploy, but:
 - Data loss : 0.01% to 10%
 - Currently no security
 - Lack of scalability
- New transport to be released in July.

Transport

- TCP: direct connection to server not scalable.
- Possible solution: use of proxies
- Proposed transport mechanism: pull data, file transfer, proxy collector.

Transport



Transport

Repository

Proxy

MSA A

FILE A1
FILE A2

MSA B

FILE B1
FILE B2

Transport

Repository

Proxy

File P1: FILE A1 ; FILE A2

GET *

MSA A

FILE A1
FILE A2

MSA B

FILE B1
FILE B2

Transport

Repository

Proxy

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2

GET *

```
graph TD; MSA_B[MSA B] -- GET * --> Proxy[Proxy];
```

MSA A

FILE A1
FILE A2

MSA B

FILE B1
FILE B2

Transport

Repository

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2



Proxy

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2

MSA A

FILE A1
FILE A2

MSA B

FILE B1
FILE B2

Transport

Repository

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2



DEL P1

Proxy

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2

MSA A

FILE A1
FILE A2

MSA B

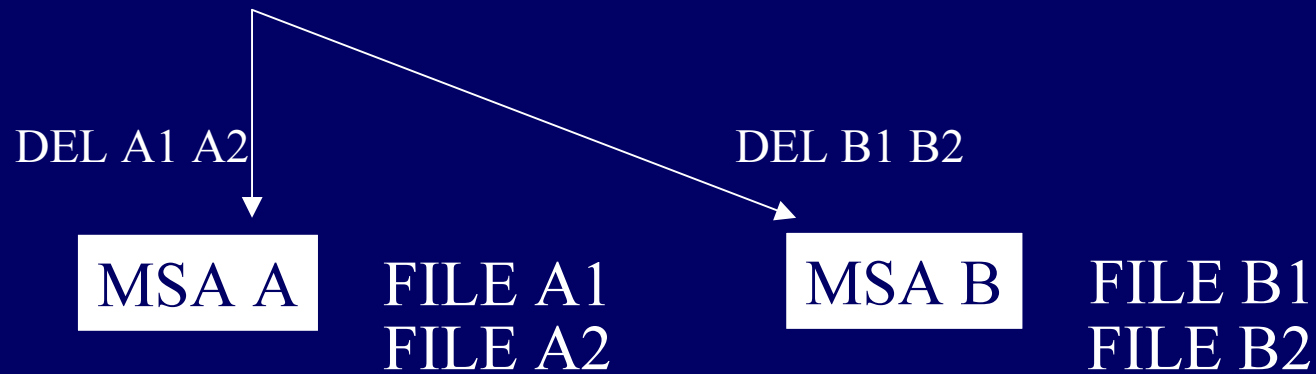
FILE B1
FILE B2

Transport

Repository

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2

Proxy



Transport

Repository

File P1: FILE A1 ; FILE A2 ; FILE B1 ; FILE B2

Proxy



Parse
and
store

MSA A

MSA B

Transport

- Transport safe
- Duplication of transport possible
 - Need a file index. Repository only keeps highest index. File already inserted in DB are rejected.
- Can be implemented with FTP.
 - But then polling required.
- Push/Pull possibles

Transport

- Open issue : security
 - Possible use of Grid-FTP
 - Other solutions also easy to implement
 - Tunneling, or use of wide spread libraries
 - Anyway, key distribution?

 Are there any preferred method?

Work to be done

- Complete work in progress
 - Architecture defined, but implementation choices to be tested and clarified
- Integration:
 - Lcfg object
 - Sensors : what do we need?