VoiCE OF THE CLIENT
Helping to influence the future
of software and systems delivery
Rational. software

Rational. software
IBM Rational software | Client event
VOICE OF THE CLIENT
Helping to influence the future
of software and systems delivery

Accelerating IBM Rational ClearCase Snapshot Views

Derek Lau
eBay
dlau@ebay.com

Let's build a smarter planet.

A Word of Warning

- Uncharted territory ahead!
- Not an IBM supported configuration
- You should be comfortable writing/debugging your own tools

The Problem

- Snapshot views load each load rule SEQUENTIALLY
  - Load <loadrule1>
  - Then load <loadrule2>
  - Then load <loadrule3>
  - etc
- There is only ONE single-threaded view_server process for each view
  - Unused CPU capacity, especially with multiple core CPUs
  - Unused network bandwidth

The Solution

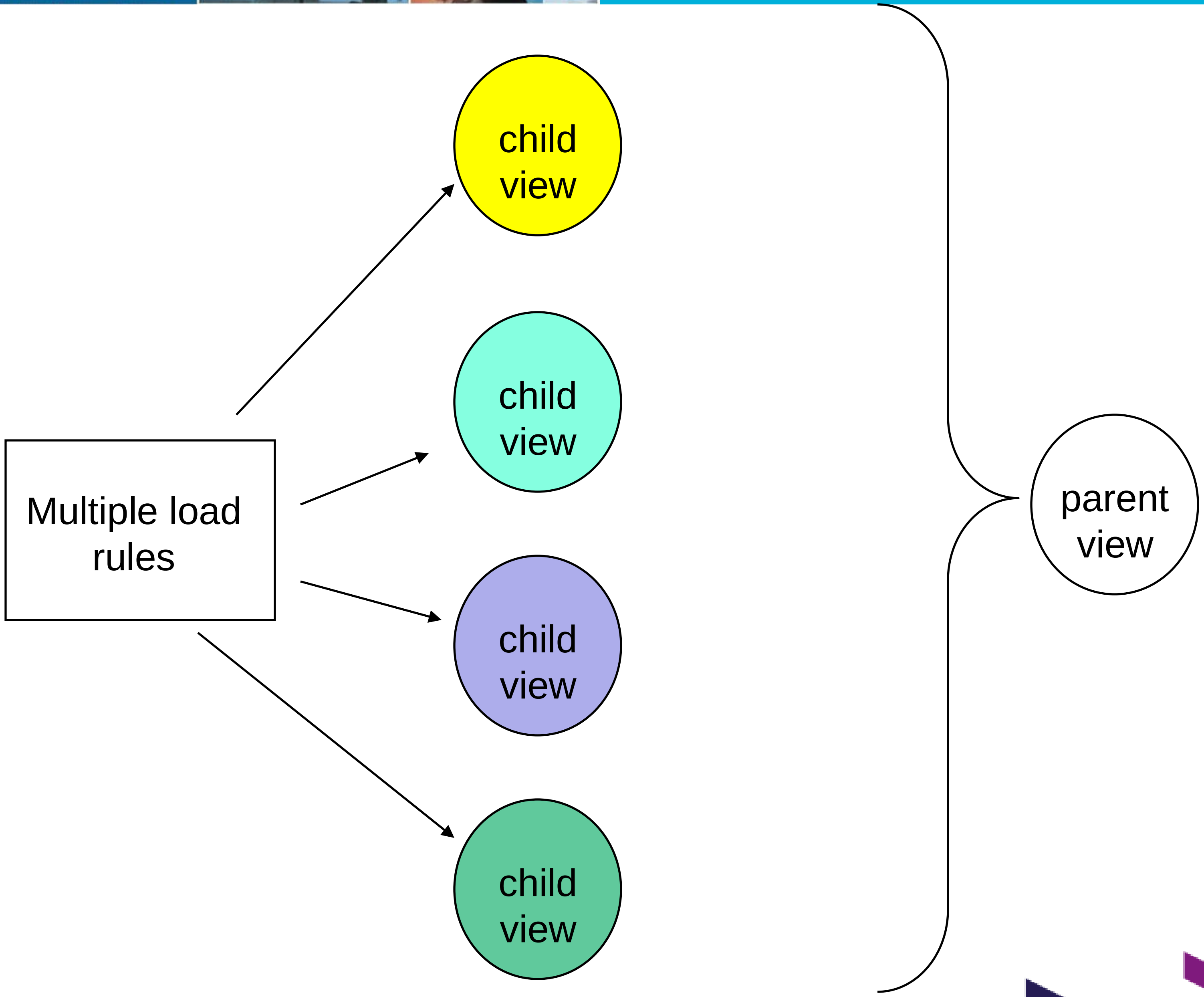
- Create X “child” views

- Each child view will process a subset of the parent’s load rules

- Sample child view config spec

```
element * CHECKEDOUT
element * .../m_dlau/LATEST
element * /main/LATEST -mkbranch m_dlau
load /TeamSCM/ClearCase
```

- Combine the workspace, config spec, .compiled_spec, db into the parent view



Multiple load rules

child view

child view

child view

child view

parent view

A Typical Snapshot View

- Typical snapshot view config spec

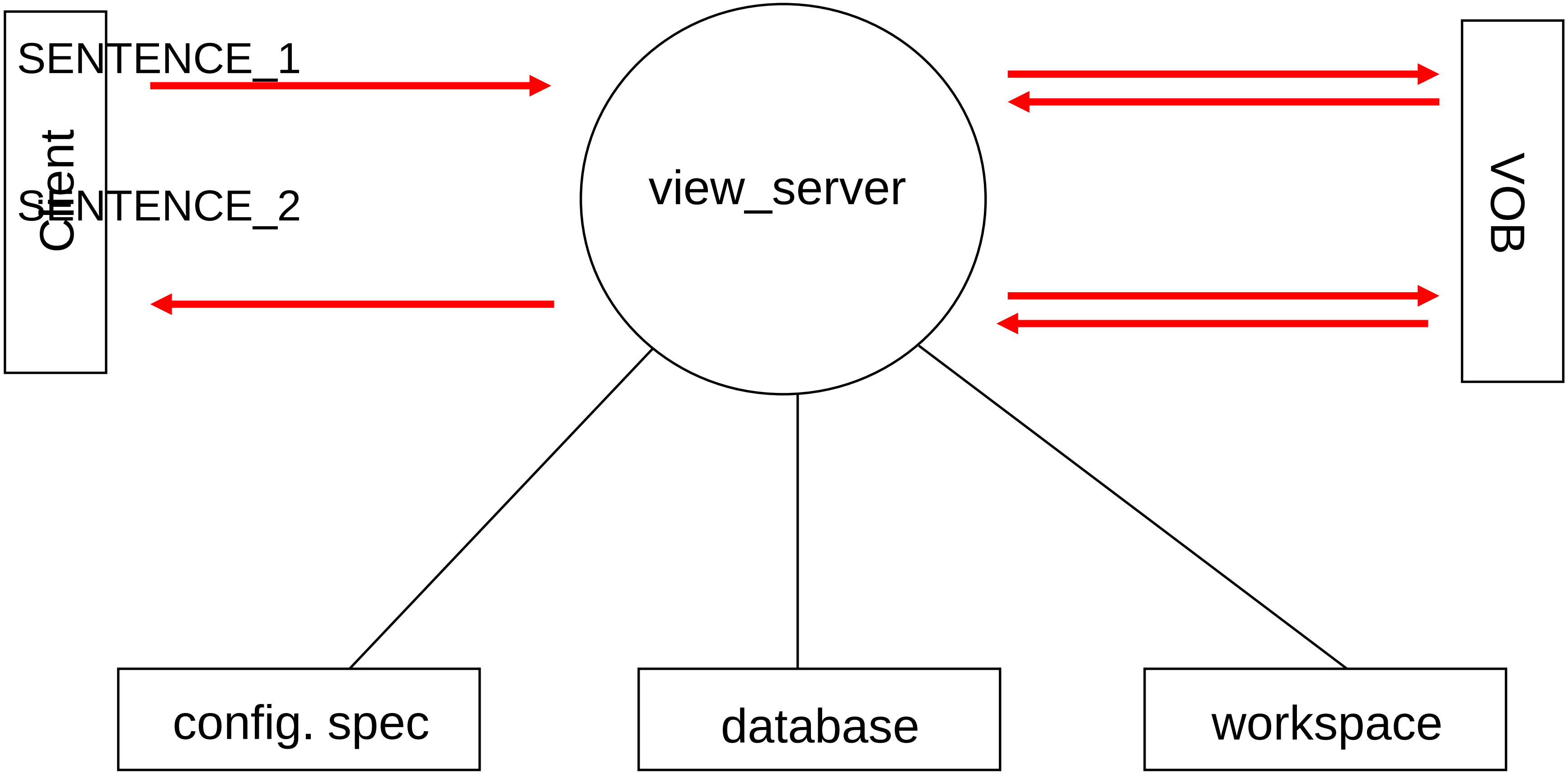
```
element * CHECKEDOUT
element * .../m_dlau/LATEST
element * /main/LATEST -mkbranch m_dlau
load /TeamSCM/ClearCase
load /TeamSCM/Admin
load /ICE/ICEConfig/scripts
... many other load rules ...
```

Technical Overview of a Snapshot View (1 / 3)

- Snapshot view workspace
  - identified by a view.dat file in the workspace directory root
- View storage (.vws or .stg)
  - config_spec
  - .compiled_spec
  - db directory
  - ... and other files
- view_server process
  - communicates with the VOB via vobrpc_server process
  - single threaded by design
  - updates the view database continuously during a load

Technical Overview of a Snapshot View (2 / 3)

- SENTENCE_1
- SENTENCE_2



Client

view_server

VOB

config. spec

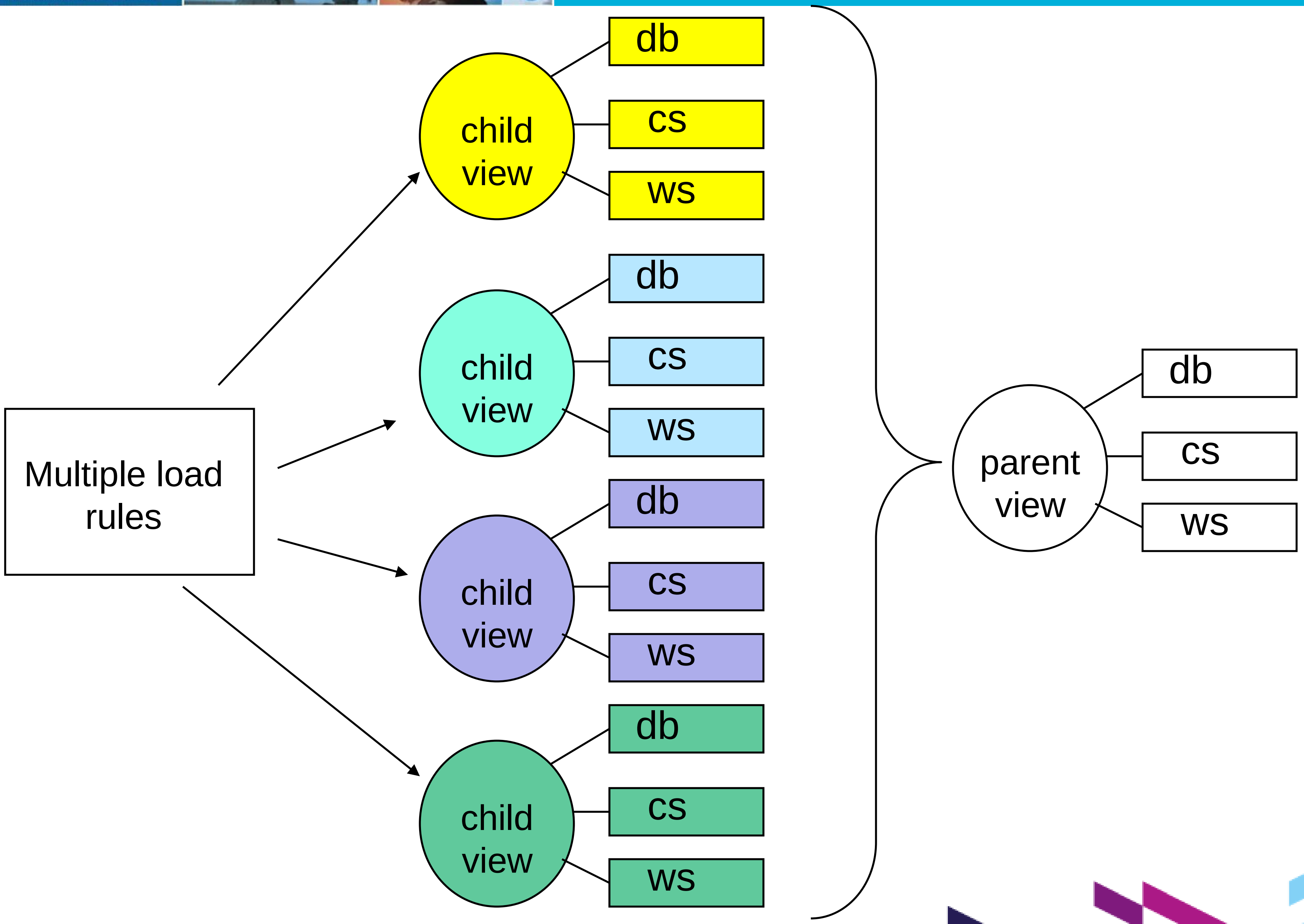
database

workspace

Technical Overview of a Snapshot View (3 / 3)

- What happens when you do setcs/edcs in a snapshot view?

- config_spec file compiled by view_server process to .compiled_spec
- each load rule is processed sequentially
  - view_server determines what version of this file/directory by parsing compiled_spec
  - view_server downloads one file
  - view_server update the view’s database
  - view_server process next file



db
cs
ws
child view
db
cs
ws
child view
Multiple load rules
db
cs
ws
child view
db
cs
ws
child view
parent view
db
cs
ws

Reassembling the View Workspace

- Copy the files from the child workspaces into the parent workspace
- If we are only doing a build, we are done!
- If we need to do ClearCase operations (eg checkin, checkout), a few more steps...

Reassembling the config_spec file

- Simple human readable text file format

```
element * CHECKEDOUT
element * .../m_dlau/LATEST
element * /main/LATEST -mkbranch m_dlau
load /TeamSCM/ClearCase
load /TeamSCM/Admin
load /ICE/ICEConfig/scripts
```

Reassembling the .compiled_spec file (1 / 5)

- Compiled spec file for our example config spec looks like this:

```
12
dfe36c75.23684e37.ab97.9f:1f:b4:6d:7c:97
6
1 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 0 0 0: 0: 0: 5 0:
2 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 0 0 0: 0: 0: 1
        17:.../m_dlau/LATEST
3 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 1 0 0: 6:m_dlau
0: 1    12:/main/LATEST
4 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 18:/TeamSCM/ClearCase
0:
5 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 14:/TeamSCM/Admin 0:
6 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 22:/ICE/ICEConfig/scripts
0:
```

Reassembling the .compiled_spec file (2 / 5)

- Human readable text file, but more complicated format

- Three main portions
  - Header
  - Element selection rules
  - Load rules

Reassembling the .compiled_spec file (3 / 5)

- Header

  12

  dfe36c75.23684e37.ab97.9f:1f:b4:6d:7c:97

  6

- Note:
  - Retain Line 1 from the parent view
  - Line 2 is the view's uuid – use parent view's uuid
  - Line 3 is the total number of element selection + load rule lines

Reassembling the .compiled_spec file (4 / 5)

- Element selection rules

```
1 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 0 0 0: 0: 0: 5 0:
2 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 0 0 0: 0: 0: 1 17:.../m_dlau/LATEST
3 1 1:* 00000000.00000000.0000.00:00:00:00:00:00 0 0 0 0 0 0 0 1 0 0: 6:m_dlau 0: 1 12:/main/LATEST
```

- Corresponding to these lines

```
element * CHECKEDOUT
element * .../m_dlau/LATEST
element * /main/LATEST -mkbranch m_dlau
```

- These lines should be the same in every child view

Reassembling the .compiled_spec file (5 / 5)

- Load rules

```
4 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 18:/TeamSCM/ClearCase 0:
5 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 14:/TeamSCM/Admin 0:
6 11 0: 00000000.00000000.0000.00:00:00:00:00:00 0 0 22:/ICE/ICEConfig/scripts 0:
```

- Corresponding to these lines

```
load /TeamSCM/ClearCase
load /TeamSCM/Admin
load /ICE/ICEConfig/scripts
```

- Notes
  - Copy the load rule line from child to parent
  - First number of each line is the line number
  - Remember to change the line number after copying!

Reassembling the db directory – view database (1 / 3)

- Run “cleartool reformatview –dump” on each view (child and parent)
  - Creates view_db.dump_file in db directory

- Sample view_db.dump_file

```
Ver 5 0 4d911418
FreezeState 1 1
Wso dde39297.34c811d9.8094.00:01:83:08:d6:9e
    ae00b94a.49ee4d03.86e0.ad:9f:54:2f:2f:52
    41:TeamSCM\ClearCase\Tools\crpview02\clients ... <snip>
Wso ...
Wso ...
...<etc>...
Eof
```

Reassembling the db directory – view database (2 / 3)

Wso lines look something like this:

```
Wso dde39297.34c811d9.8094.00:01:83:08:d6:9e
  ae00b94a.49ee4d03.86e0.ad:9f:54:2f:2f:52
  41:TeamSCM\ClearCase\Tools\crpview02\clients
  28257433.276940d7.937f.1f:34:17:8d:e9:44
  f33f38a8.bea44e36.9bfc.a7:d9:4a:89:89:20 8:index.pl
  00000000.00000000.0000.00:00:00:00:00:00
  00000000.00000000.0000.00:00:00:00:00:00
  00000000.00000000.0000.00:00:00:00:00:00 170b 1 4b95a2f5 0 0 3 8f
```

Second last number of each Wso line analogous “Rule: ...” part of “cleartool ls”

```
element * CHECKEDOUT
element * .../m_dlau/LATEST
element * /main/LATEST -mkbranch m_dlau
```

Reassembling the db directory – view database (3 / 3)

- Reassemble db file
  - Create the headers (“Ver...” and “FreezeState...” lines) by copying from one of the views
  - Concatenate all “Wso...” lines from child view dump file to parent’s dump file
  - Append “Eof” marker
- Reload database by running “cleartool reformatview –load” on the parent view
- Parent db now contains loaded file info for all files loaded in all child views
- Remove child views when done (cleartool rmview -force)

eBay’s Use Case

- Standard developer desktop
  - 2 dual core CPUs (Xeon)
  - 8GB RAM
  - 100Mb ethernet
- View workspace size ~ 5GB
- Across ~100 load rules / ~30 VOBs
- Takes about 45 minutes to create a snapshot view from scratch

Performance Metrics

- Case #1 – default single threaded snapshot view load, using local SATA disk
  - Throughput = 2 MBps , Time = 45 minutes
  - Bottleneck? Sequential single-threaded design
  - Solution? Implement parallel design
- Case #2 – parallel snapshot view load, using local SATA disk
  - Throughput = 5 MBps, Time = 20 minutes
  - Bottleneck? Disk IO
  - Solution? Use SSD drives.
- Case #3 – parallel snapshot view load, using SSD
  - Throughput = 8 MBps, Time = 10 minutes
  - Bottleneck? Possibly network
  - Solution? Work in progress...

Extending the Idea

- Similar technique can be used to replicate entire views, but NOT view privates
- Use Case
  - Lead dev loads his snapshot, does a verification build, rebuild jars etc
  - Everyone else uses a clone of his view as a starting point for development.
- BEFORE (loading view using SSD disk)
  - Time = 10 minutes to load view, 60+ minutes to do initial build
- AFTER (cloning the view)
  - Time = 12 minutes

Considerations and Other Thoughts

- Assumes no symbolic links and non-UCM config spec
- Your Mileage May (WILL!) Vary ☺

www.ibm/software/rational

https://www14.software.ibm.com/iwm/web/cc/earlyprograms/rational.shtml

www.ibm/software/rational

https://www14.software.ibm.com/iwm/web/cc/earlyprograms/rational.shtml

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.