Lustre & SELinux: in theory and in practice

Septembre 22nd, 2014

Sebastien Buisson

Parallel File Systems
Extreme Computing R&D

Lustre & SELinux

- Lustre on an SELinux-enabled client
- Security policy enforcement on Lustre
- Impact of security over performance

Lustre on an SELinux-enabled client

Lustre on an SELinux-enabled client

From Lustre Operations Manual

*Before installing the Lustre software, make sure you disable Security-Enhanced Linux (SELinux) on all Lustre **servers** and **clients**. The Lustre software does not support SELinux. Therefore, the SELinux system extension must be disabled on **all** Lustre nodes.*

BUT

- since LU-506 (Lustre 2.3), replace `ll_d_add()` with `d_add()`
  - internally calls `security_d_instantiate()`
- SELinux does not prevent Lustre from working properly anymore

Security policy enforcement on Lustre

Security policy enforcement on Lustre

To begin with:

- Create a file and see its security attributes:

```
[nodeA]# ls -lZ /lustre/file1
-rwxr-xr-x. root root system_u:object_r:default_t:s0   /lustre/file1
```

- But in MDT:

```
# debugfs /dev/ram1 -R "stat ROOT/file1"
Inode: 1048584   Type: regular    Mode:  0666   Flags: 0x0
Size of extra inode fields: 28
Extended attributes stored in inode body:
 lma = "00 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 05 00 00 00 00 00 00 00 " (24)
 lma: fid=[0x200000402:0x5:0x0] compat=0 incompat=0
 lov = "d0 0b d1 0b 01 00 00 00 05 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 00 00 10 00 01
        00 00 00 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 " (56)
 link = "df f1 ea 11 01 00 00 00 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 17 00 00 00
         02 00 00 00 07 00 00 00 01 00 00 00 00 66 69 6c 65 31 " (47)
```

⇒ no xattr containing security information

- Shows need to store security information permanently

Security policy enforcement on Lustre

Code evolutions proposed in LU-5560

- initialize security context in `struct inode`

  `security_inode_init_security()`

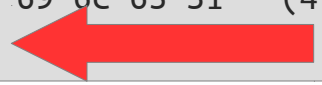
- store it in security.selinux xattr

  `ll_setxattr()`

With the patch:

```
[nodeA]# ls -lZ /lustre/file1
-rwxr-xr-x. root root system_u:object_r:file_t:s0   /lustre/file1
```

```
# debugfs /dev/ram1 -R "stat ROOT/file1"
Inode: 1048584   Type: regular    Mode:  0666   Flags: 0x0
Size of extra inode fields: 28
Extended attributes stored in inode body:
 lma = "00 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 05 00 00 00 00 00 00 00 " (24)
 lma: fid=[0x200000402:0x5:0x0] compat=0 incompat=0
 lov = "d0 0b d1 0b 01 00 00 00 05 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 00 00 10 00 01
        00 00 00 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 " (56)
 link = "df f1 ea 11 01 00 00 00 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 17 00 00 00
         02 00 00 00 07 00 00 00 01 00 00 00 00 66 69 6c 65 31 " (47)
 selinux = "system_u:object_r:file_t:s0\000" (31)
```



Security policy enforcement on Lustre

SELinux user « unconfined »

- transparent access to Lustre filesystem

Specific SELinux user

- without dedicated security policy: all accesses denied
- with a specific security policy defined on the Lustre client: access granted in accordance with the policy

Security policy extract

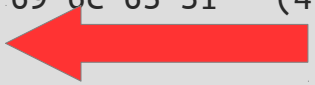
```
allow user_t file_t:file { getattr open read write create unlink };
allow user_t file_t:dir { search getattr open read write add_name create remove_name rmdir };
```

Security policy enforcement on Lustre

- Security context change from another node

```
[nodeB]# chcon -t home_t /lustre/file1
[nodeB]# ls -lZ /lustre/file1
-rwxr-xr-x. root root system_u:object_r:home_t:s0   /lustre/file1
```

```
# debugfs /dev/ram1 -R "stat ROOT/file1"
Inode: 1048584   Type: regular    Mode:  0666   Flags: 0x0
Size of extra inode fields: 28
Extended attributes stored in inode body:
 lma = "00 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 05 00 00 00 00 00 00 00 " (24)
 lma: fid=[0x200000402:0x5:0x0] compat=0 incompat=0
 lov = "d0 0b d1 0b 01 00 00 00 05 00 00 00 00 00 00 00 02 04 00 00 02 00 00 00 00 00 10 00 01
        00 00 00 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 " (56)
 link = "df f1 ea 11 01 00 00 00 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 17 00 00 00
         02 00 00 00 07 00 00 00 01 00 00 00 00 66 69 6c 65 31 " (47)
 selinux = "system_u:object_r:home_t:s0\000" (31)
```



- But on initial node:

```
[nodeA]# ls -lZ /lustre/file1
-rwxr-xr-x. root root system_u:object_r:file_t:s0   /lustre/file1
```

- Shows need for security information coherency
  - but this is a very special case

Security policy enforcement on Lustre

Code evolution proposed in LU-5560

- drop inode from cache after use

```
struct super_operations lustre_super_operations =
{
<<<snip>>>
        .drop_inode     = generic_delete_inode,
<<<snip>>>
};
```

- discussions about the viability of this proposal...

Impact of security over performance

Impact of security over performance

Tests

- Environment
  - 1 client node, 1 Lustre server MDS+OSS, ramdisk storage
- Benchmark
  - mdtest
  - « directory per process » mode

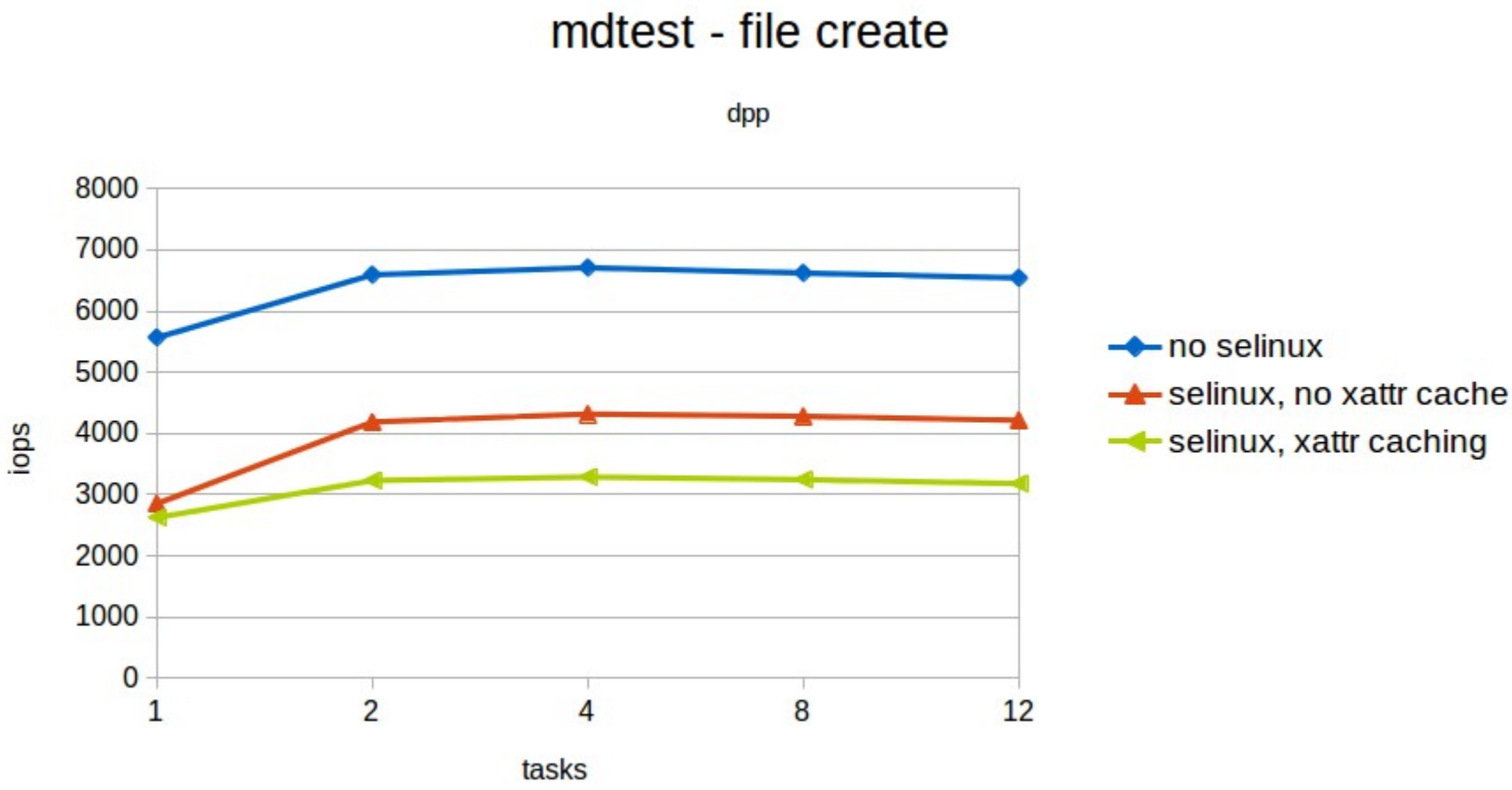
Code tested

- Permanent security info, without coherency
  - Lustre 2.5.2 + « security.selinux xattr » patch

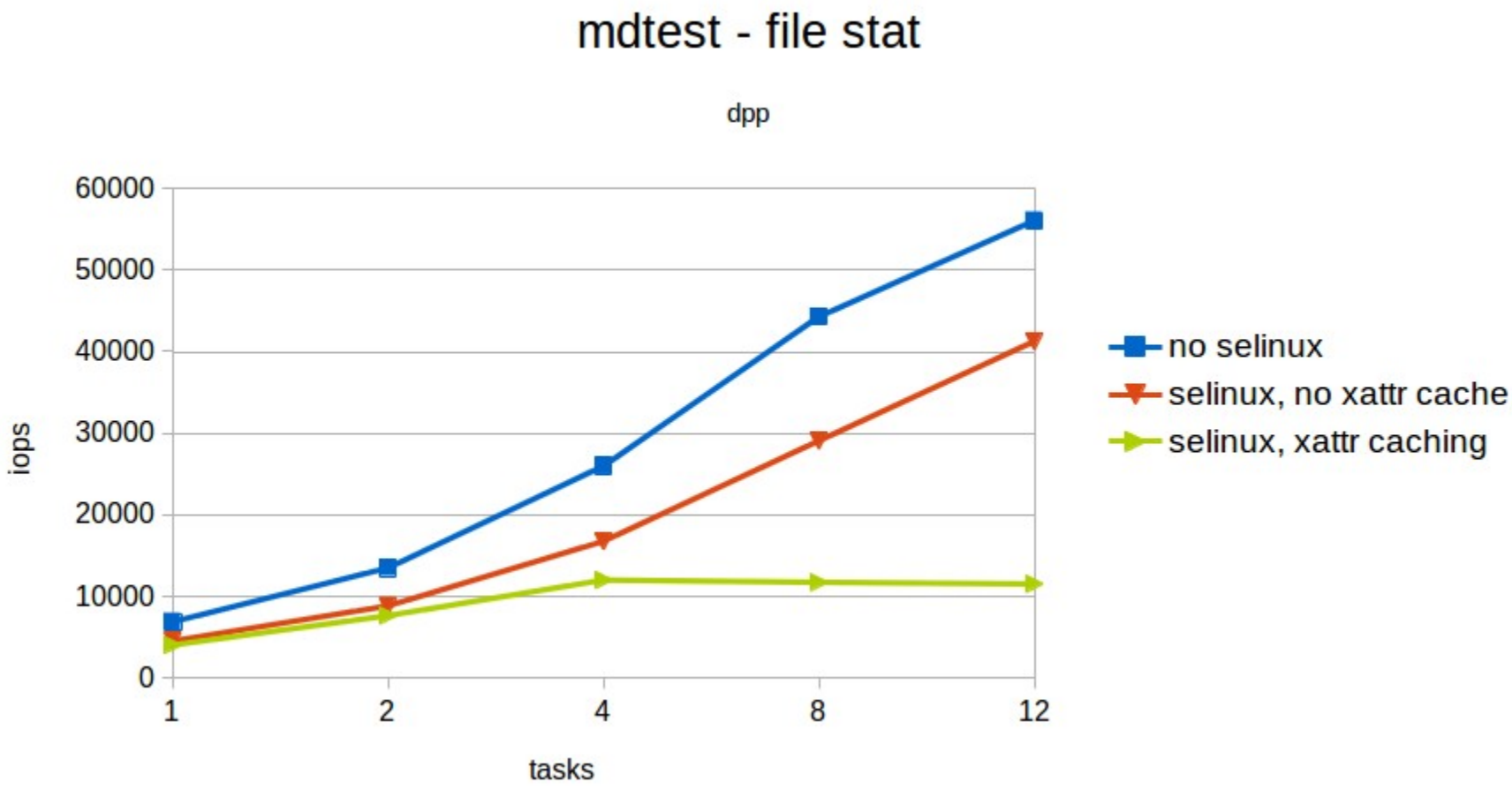
Objectives

- SELinux impact over metadata performance
- “xattr caching” benefit

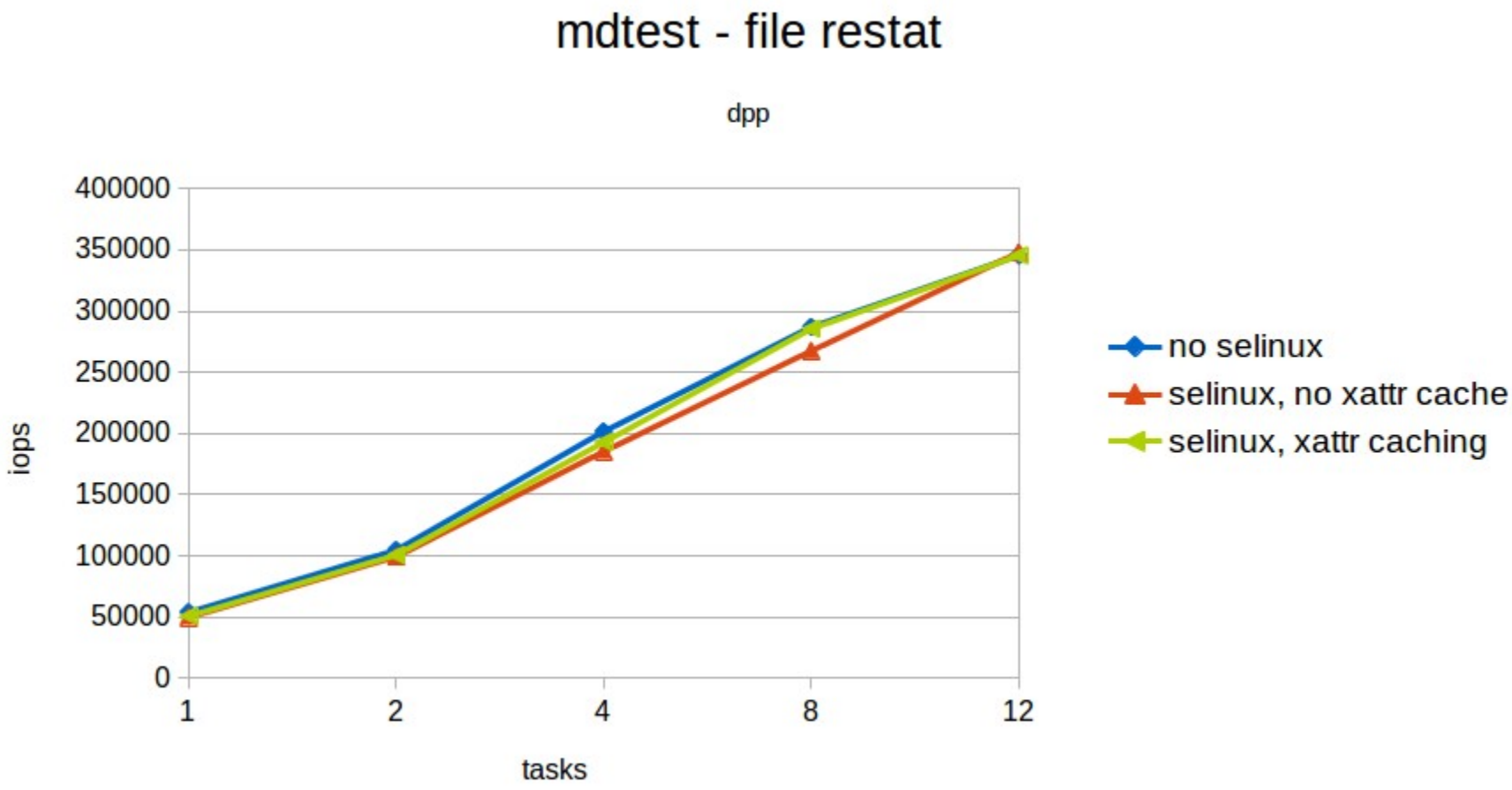
mdtest - create



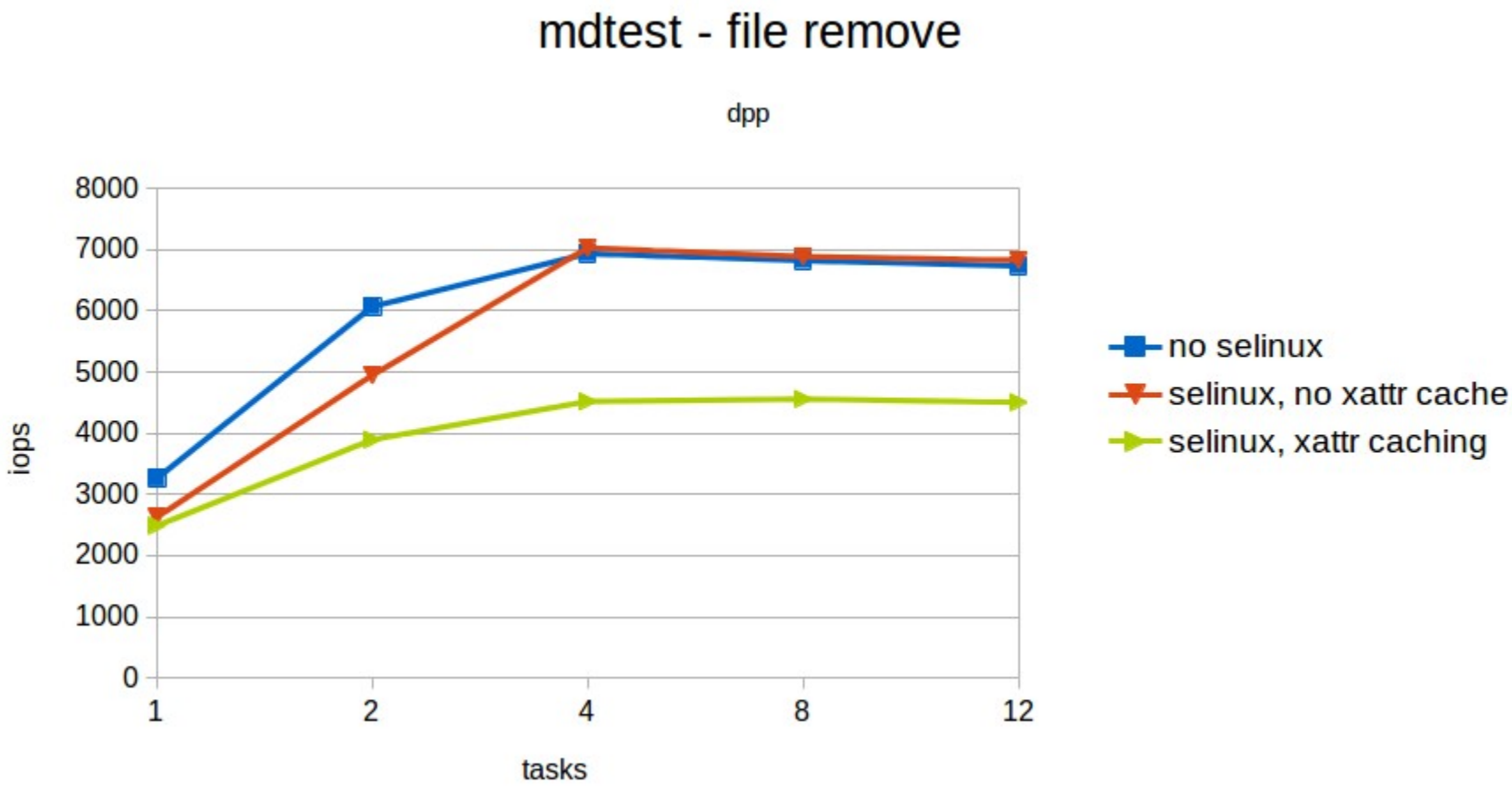
mdtest - stat



mdtest - restat



mdtest - remove



mdtest

Tests summary:

Lustre 2.5.2 + « security.selinux xattr » patch

Difference from « no selinux »	selinux no xattr caching	selinux xattr caching
create	- 35 %	- 50 %
stat	- 35 %	- 75 %
restat	*similar*	*similar*
remove	*similar*	- 35 %

Impact of security over performance

Code tested

- coherency in addition to permanent security info
  - previous code + inode drop after use

Objectives

- viability of the solution

mdtest

Tests summary:

Lustre 2.5.2 + « security.selinux xattr » patch + inode drop after use

	Difference from « no selinux »
create	-40 %
stat	-85 %
restat	-95 %
remove	-25 %


Bull
an atos company