Integrare AFS e GPFS in modo trasparente per l'utente: una applicazione di AFS OSD

Hartmut Reuter
reuter@rzg.mpg.de

Introduzione

AFS OSD o col nome completo “OpenAFS + Object Storage” è stato sviluppato anche con supporto di ENEA tra 2005 e 2007.

Adesso è in produzione al RZG da due anni.

Nei ultimi anni una idea piu vecchia di usare *shared filesystems* per AFS è stata realizzata in AFS OSD

in *cluster Linux* con GPFS, AFS può sfruttare la velocità di GPFS

e i dati rimangono visibili non solo nel cluster ma dappertutto, non solo su Linux ma su ogni piattaforma

CRESCO a Portici sta facendo una installazione di prova di questa soluzione

Quindi spiegherò come funziona questa integrazione e come la si usa

Parlerò anche un po' delle altre possibilità offerte da AFS OSD

GPFS e AFS: una coppia bella

GPFS e Lustre sono molto piu veloci di AFS specialmente in un ambiente di rete velocissima come Infiniband o 10GE.

Ma GPFS e Lustre hanno anche i loro limitazioni:

- Con una *block size* grande non sono efficienti per *file* piccoli
- Creazione e cancellazione di *file* sono abbastanza lente
- Non c'è un modo sicuro per la esportazione al WAN o neanche al *desktop*
- Sono montabili solo su Linux (o AIX in caso di GPFS)

Queste sono caratteristiche complementari a quelle di AFS

- che è molto bravo coi *file* piccoli
- che permette un accesso sicuro su tutto il mondo e su quasi tutte le piattaforme

Allora: perché non sposare AFS e GPFS per combinare le loro virtù?

Come combinare AFS e GPFS?

GPFS (o Lustre) si monta solo nei *cluster* affidabili che tipicamente sono connessi con reti velocissime.

Permettiamo che all'interno del *cluster* AFS usi GPFS (o Lustre)

- da parte del *server* AFS usando GPFS per la partizione dove i file vengono immagazzinati
- da parte del *client* AFS facendo sì che il *cache manager* possa scrivere o leggere i file direttamente col protocollo di GPFS (o Lustre)
- Per l'utente i *file* vivono solo nel albero di /afs (perché l'utente non privilegiato non ha accesso al *directory* sotto il quale sono immagazzinati i *file* nel GPFS).
- L'utente ha bisogno di un *token* AFS e gli ACL dei *directory* AFS rimangono in vigore.

Su tutte altre macchine (fuori del *cluster*) l'accesso a questi *file* avviene tramite i *server* di AFS con il protocollo tradizionale.

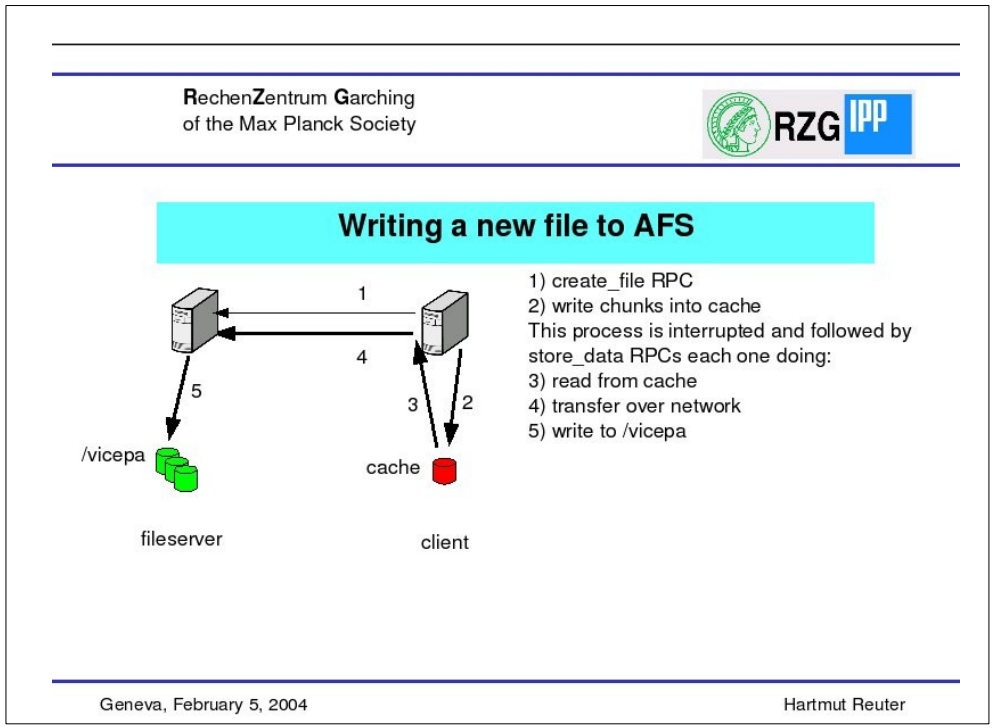
- Così i *file* in GPFS diventano accessibili su tutto il mondo in modo sicuro, anche sotto Windows connesso con ADSL a casa vostra!

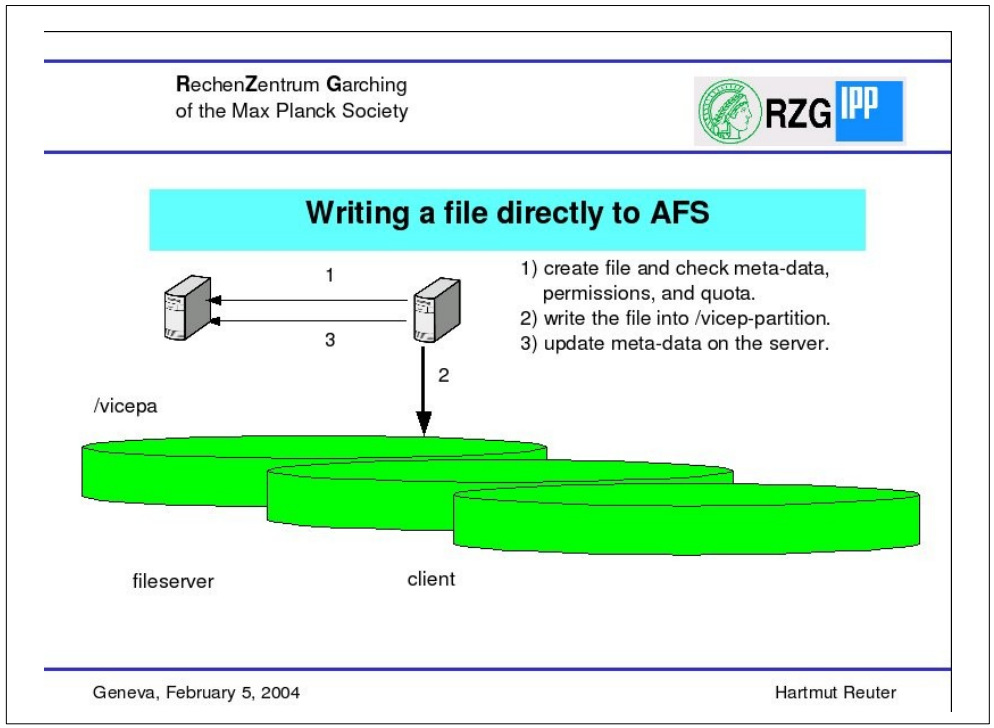
Accesso diretto dal cliente

Questa idea non è nuova:

già sei anni fa avevo proposto al CERN di usare *shared file systems* per AFS

La HEPIX Storage Group ha verificato nel 2008 e 2009 che questa tecnica funziona efficacemente

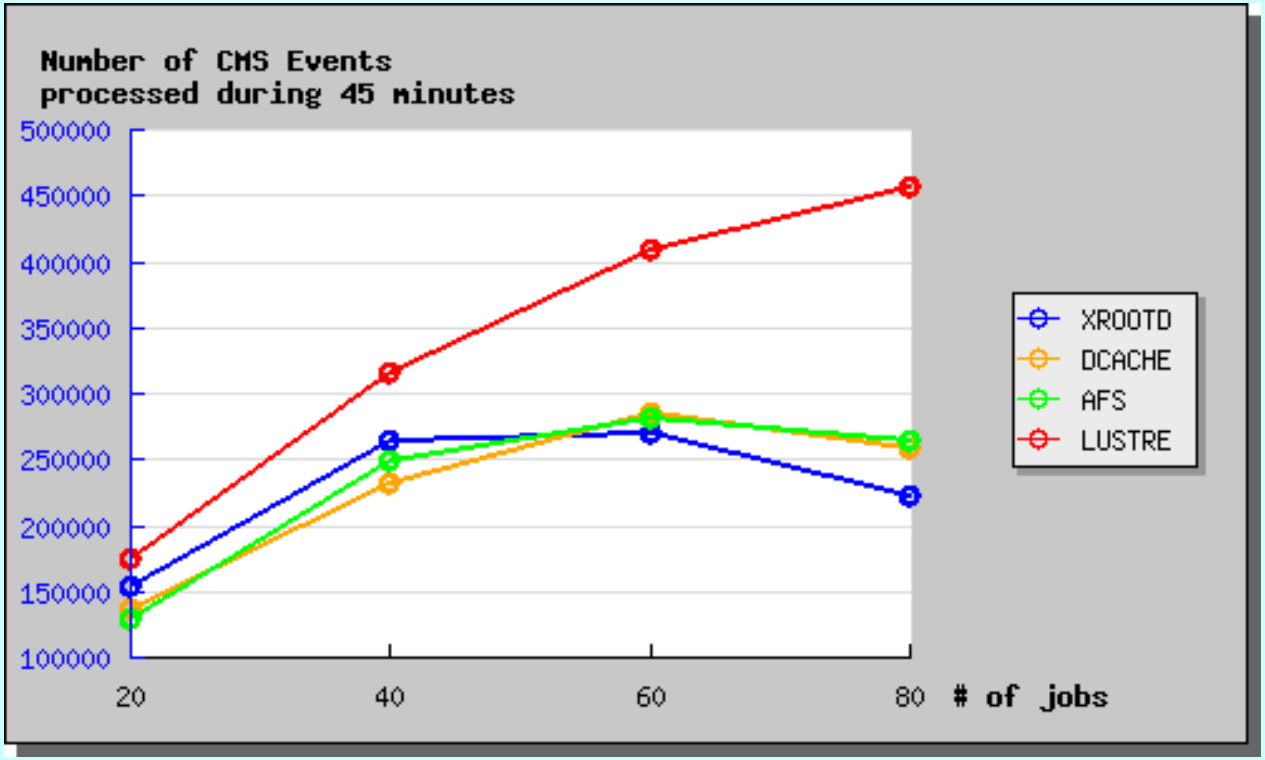




HEPIX-Tests (1)

The "HEPIX Storage Working Group" developed 2008 a use case for distributed storage systems based on CERN's soft- and middleware stack for CMS

In a 1st round in 2008 at FZK (Forschungszentrum Karlsruhe) Andrei Maslennikov compared: AFS, DCACHE, LUSTRE, and XROOTD.

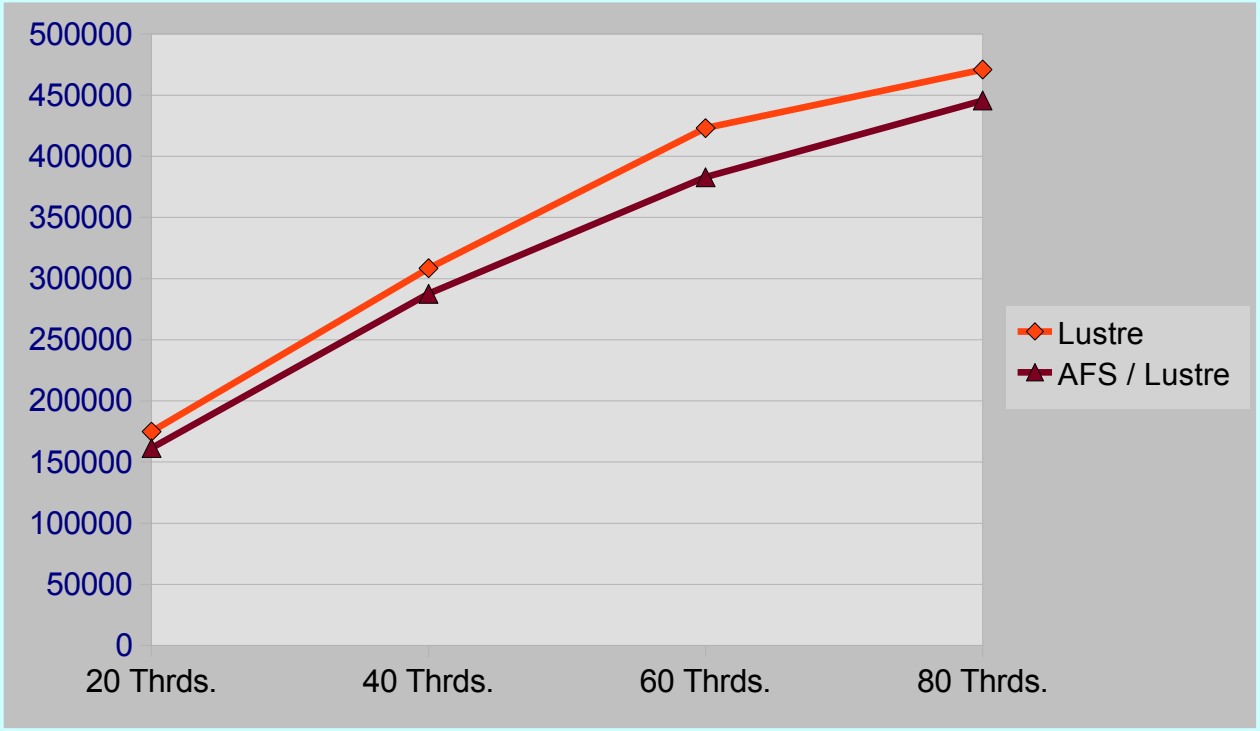


Lustre performs much better than AFS, DCACHE and XROOTD

Source: „HEPIX storage working group, - progress report, 2.2008-", Andrei Maslennikov, Taipei October 20, 2008

HEPIX-Tests (2)

In June 2009 again at FZK, but with different server hardware only Lustre and in AFS embedded Lustre were compared.



Source: SMS from Andrei Maslennikov , June, 3,2009

AFS with embedded Lustre comes close to Lustre native.

Evitare file piccoli nel GPFS

Come detto prima:

un svantaggio del GPFS (e Lustre) è che non è molto efficiente per i file piccoli.

Per separare i file grandi da quelli piccoli ci ha aiutato un progetto finanziato anche da ENEA:

"OpenAFS + Object Storage"

Prima di presentarlo vorrei spiegare quale erano i motivi di svilupparlo

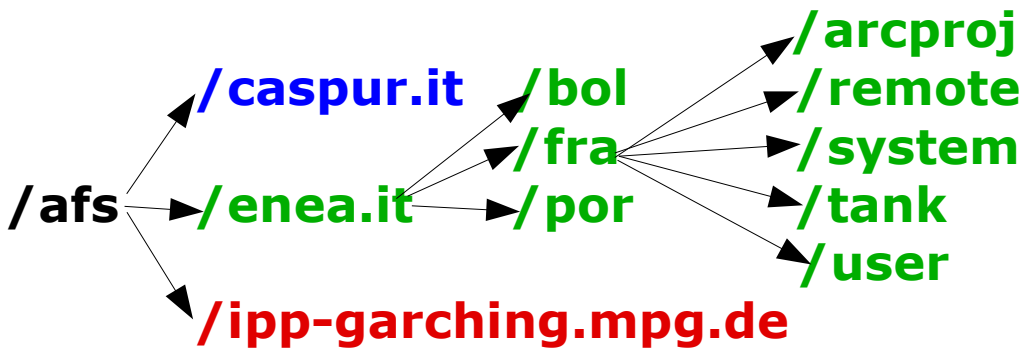
AFS come lo conoscete

AFS (Andrew File System)

- sviluppato alla Carnegie Mellon University a Pittsburgh nei anni 80
- prodotto commerciale della ditta Transarc, poi IBM
- diventato *open source* nel 2000 col nome OpenAFS

Montato sui sistemi Unix e Linux su /afs

L'albero sotto /afs si dirama nei alberi delle diverse celle amministrative



Ogni *directory* verde qui è il *mount point* di un volume AFS nella cella „enea.it".

Volumi AFS classici

Un volume AFS può contenere

- un sacco di *file* e *directory*
- fino a due TB (terabyte) di dati
- *mount points* di altri volumi AFS

Tutti i *file* e *directory* sono immagazzinati in una singola partizione di disco sul *fileserver.*

Un volume RW (*read/write)* puo esistere solo su un unico *fileserver,*

però volumi RO (*read only)* possono essere replicati su fino a 15 *fileserver.*

La replicazione

- può aumentare il *throughput* totale nel caso che molti clienti vogliono leggere *files* dallo stesso volume
- crea ridondanza per il caso che un *fileserver* sia morto.
- è usato tipicamente per *software* ma non per dati

OpenAFS + *Object Storage*

In 2004 Rainer Többicke del CERN ha avuto l'idea di combinare AFS con o*bject storage* e ha sviluppato un prototipo molto semplice.

In 2005 Andrei Maslennikov di CASPUR è riuscito con successo a creare un progetto per lo sviluppo di „OpenAFS + Object Storage“:

il progetto era finanziato per due anni da CERN, ENEA and INFN.

due programmatori giovani sono stati assunti

io dovevo accompagnare il progetto come specialista del AFS

In 2006 mi veniva l' idea che „OpenAFS + Object Storage” avrebbe potuto sostituire MR-AFS che era usato come sistema HSM da noi

Incominciavo di occuparmi attivamente alla programmazione.

In 2007 il progetto finiva con una seria di prove che dimostravano che infatti la prestazione dipende linearmente dal numero di OSD (*object storage device)* usati.

I risultati erano presentati al HEPIX spring meeting 2007

https://indico.desy.de/materialDisplay.py?contribId=16&sessionId=39,materialId=slides&confId=257

Cosa è *object storage?*

Object storage systems sono *filesystem* distribuiti che immagazinano i dati in *object storage devices* (OSD) e memorizzano i metadati in *metadata server*.

Esempi di sistemi *object storage* sono **Lustre** e **Panasas**

Accesso a un *file* in *object storage* esige i seguenti passi sul cliente:

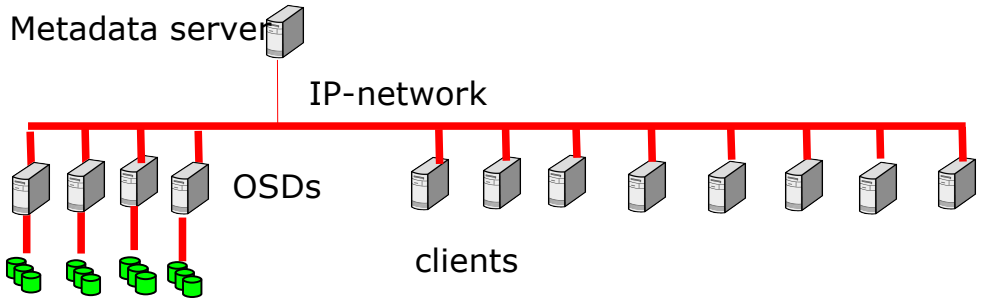
rpc al *metadata server* che verifica il diritto di permesso e ritorna un *handle* cifrato per ogni *object* del *file*.

rpc al OSD per leggere o scrivere il *object* usando il *handle* che aveva ricevuto dal *metadata server*

Il OSD è capace di dicifrare il *handle* perché ha un segreto in comune col *metadata server* . Il vantaggio di questa tecnica è che l'OSD non deve sapere nulla di utenti e diritti di accesso.

Un *file* può anche essere *striped* (distribuito) o *mirrored* (duplicato) su multipli OSD

:



Scopo (primario) di AFS OSD

Abbiamo sviluppato "OpenAFS + Object Storage" o breve "AFS OSD"

per distribuire i *file* di un volume tra server multipli

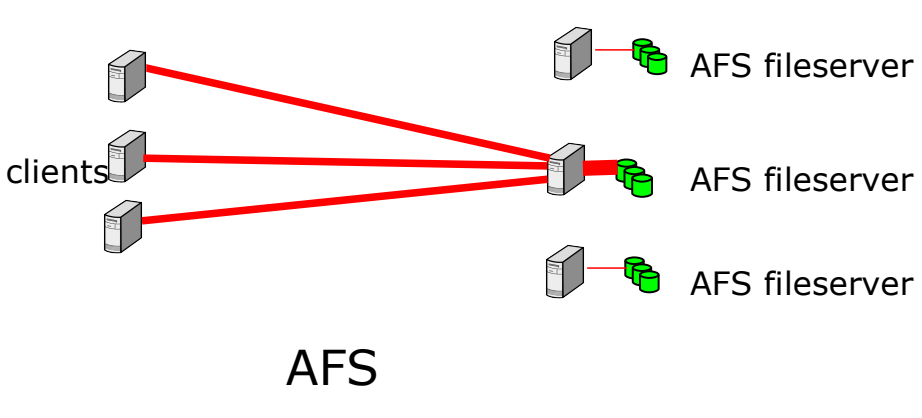
al **fileserver** per i *file* piccoli

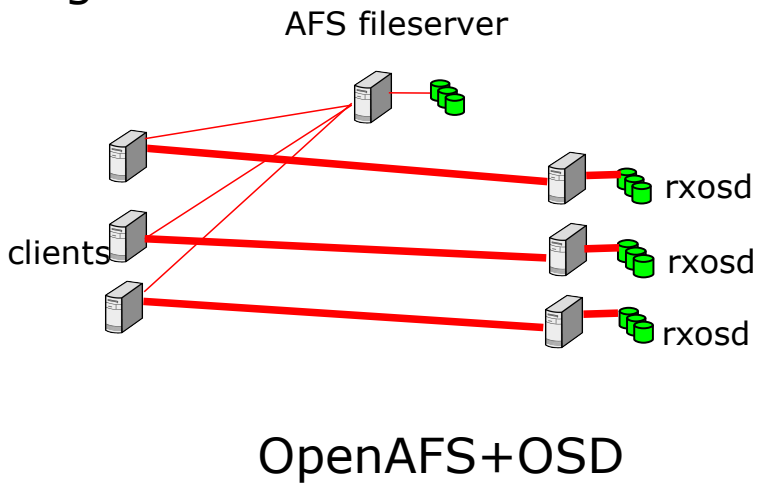
a una serie di **rxosd** per i *file* grandi

per permettere *striped files* e/o *mirrored files* anche in AFS

Esempio: 3 clienti accedono 3 file nello stesso volume AFS

Il carico si distribuisce meglio con AFS OSD





"OpenAFS + Object Storage"

Estensione di OpenAFS che permette immagazzinare *file* in OSDs.

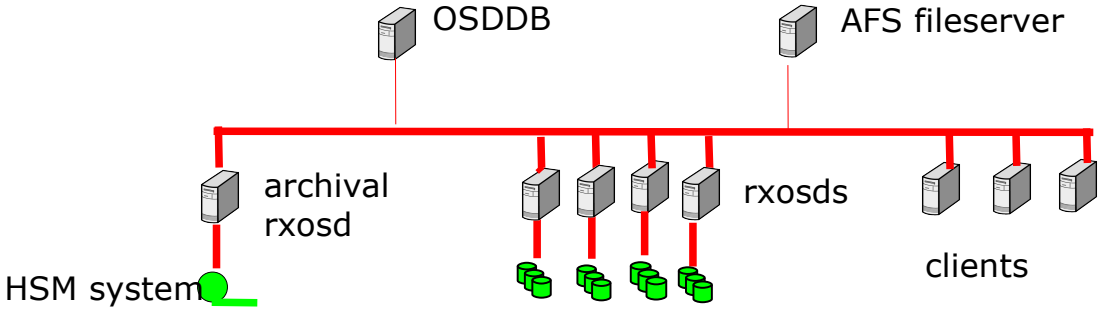
OSDs (object storage devices) di OpenAFS usano il protocollo RX e sono chiamati dunque "rxosd".

Il *fileserver* memorizza i metadati dei *file* che sono immagazzinati nel *object storage*.

Un nuovo *database* AFS chiamato OSDDB che sa quali OSD esistono nella cella e quale caratteristiche hanno.

Un *file* in *object storage* puo essere semplice o *striped/mirrored* su più di un OSD e può avere anche copie archiviate su *archival rxosds* (tipicamente sistemi HSM).

Le copie archiviate su nastro offrono la possibilità di usare AFS come sistema HSM: cancellare *file* inattivi dai OSD



Esempio per l'uso degli OSD


```
  1 MB -   2 MB         0   0.00  14.33         0  B   0.00   0.00
  2 MB -   4 MB         0   0.00  14.33         0  B   0.00   0.00
  4 MB -   8 MB         1   0.03  14.36     4.747 MB   0.00   0.00
  8 MB -  16 MB         3   0.08  14.43    40.959 MB   0.00   0.00
 16 MB -  32 MB        34   0.85  15.29   606.325 MB   0.03   0.04
 32 MB -  64 MB         9   0.23  15.51   463.253 MB   0.03   0.06
 64 MB - 128 MB        45   1.13  16.64     4.055 GB   0.23   0.29
128 MB - 256 MB        38   0.95  17.60     6.862 GB   0.39   0.68
256 MB - 512 MB       234   5.87  23.47    74.095 GB   4.17   4.85
  1 GB -   2 GB         0   0.00  99.97         0  B   0.00  94.79
  2 GB -   4 GB         0   0.00  99.97         0  B   0.00  94.79
  4 GB -   8 GB         0   0.00  99.97         0  B   0.00  94.79
  8 GB -  16 GB         0   0.00  99.97         0  B   0.00  94.79
 16 GB -  32 GB         0   0.00  99.97         0  B   0.00  94.79
 32 GB -  64 GB         0   0.00  99.97         0  B   0.00  94.79
 64 GB - 128 GB         1   0.03 100.00    92.641 GB   5.21 100.00
--------------------------------------------------------------------
Totals:              3984 Files             1.734 TB

Storage usage:
--------------------------------------------------------------------
                1 local_disk        582 files     340.155 MB
  arch. Osd     5 tape             3402 objects     1.734 TB
        Osd     9 mpp-fs9-a         636 objects   311.112 GB
  arch. Osd    13 hsmgpfs          3402 objects     1.734 TB
        Osd    23 mpp-fs11-gj       274 objects   135.549 GB
        Osd    24 mpp-fs12-a        268 objects   134.127 GB
        Osd    25 mpp-fs13-a        273 objects   133.819 GB
        Osd    41 mpp-fs15-gi       125 objects    62.358 GB
        Osd    42 mpp-fs14-gh         4 objects    67.247 MB
        Osd    43 mpp-fs10-gk       166 objects    82.848 GB
--------------------------------------------------------------------
Total                            9132 objects     4.309 TB
```

Parte del risultato del commando „vos traverse ...“ che fa vedere una statistica dei *file* di un volume

Il volume contiene 1.73 TB

582 *file* < 1 MB nella partizione del *fileserver* (local_disk)

3402 *file* > 4 MB distribuiti su 7 *nonarchival OSD* e con copie su tutte e due *archival OSD*

Solo 1856 dei 3402 *file* grandi si trovano su dischi. Il resto è solo sui nastri dei sistemi HSM.

AFS OSD + GPFS

Adesso combiniamo GPFS e AFS OSD. Come funziona?

Se il utente vuole accedere un *file* in *object storage* il *cache manager* manda un rpc al *fileserver* per avvertire una operazione asincrona.

Il fileserver verifica il diritto del utente per la operazione intenzionata e ritorna al *cache manager* tutte le informazioni necessari per contattare il OSD

Normalmente il *cache manager* poi lancia *rpc* al OSD dove i dati sono immagazzinati

Se però il cache manager aveva trovato la partizione del OSD montato sulla macchina del cliente (riconoscibile da un bollo messo lì dal rxosd)

il *cache manager* apre il *file* direttamente e legge o scrive i dati.

In ogni caso la operazione asincrona è protetta da un *lock* per il *file* nel *fileserver* cosi che nessun altro può modificare il *file* durante questa operazione.

Alla fine il cache manager lancia un rpc al fileserver per informarlo della fine della operazione asincrona e, nel caso che il file sia stato scritto, della sua lunghezza effettiva.

Configurazione prevista per CRESCO a Portici

Parte del GPFS visibile sui *cluster* di CRESCO serve come OSD per AFS.

Una macchina dove il GPFS è montato gira il **rxosd**

Un *fileserver* AFS fornito colla versione di AFS OSD contiene volumi AFS previsti per l'uso di *object storage*:

I *file* piccoli sono immagazinati sul *fileserver.*

I *file* grandi

dentro il cluster sono scritti direttamente nel GPFS

fuori del cluster mandato al rxosd

Come sempre con AFS ma diversamente dall'uso normale di GPFS l'utente (e anche I suoi batch job) hanno bisogno di un *token* AFS.

Dal punto di vista del utente tutto succede in modo completamente trasparente:

i file immagazzinati nel GPFS si comportano come tutti altri file nel AFS

Primi Risultati a Portici

Usiamo la partizione GPFS /gpor_proj di 27 TB anche per AFS OSD.
Sul cliente usiamo 256 MB di *memory cache* con una chunk size di 1 MB

Scrivere o leggere *file* grandi per fileserver normali ha una velocità di ca. 60 MB/s

Scrivere *file* grandi tramite la tecnica nuova ha una velocità di ca. 180 MB/s

Leggere *file* grandi tramite la tecnica nuova ha una velocità di ca. 240 MB/s

Queste cifre rimangono ancora ben' sotto la velocità del GPFS nativo, ma sono un fattore 2.5 a 4 piu alti di quelli di AFS normale

Altri usi di AFS OSD

AFS OSD offre ancora altre possibilità oltre quelle previste per CRESCO a Portici:

Se la quantità dei dati cresce velocemente

basta aggiungere piu OSD senza muovere volumi da un fileserver al altro perché I dati nuovi vanno automaticamente nei OSD nuovi.

Se il sito ha un sistema HSM in operazione quello può servire anche per I file nei OSD

sia come backup

sia per migrazione di dati da disco a nastro e viceversa

Fino ad adesso AFS OSD si trova in produzione vera solo al nostro centro calcolo RZG dove la funzionalità HSM è usata intensamente

RZG e MPG

RZG (Rechenzentrum Garching) è il centro calcolo della Max-Planck-Gesellschaft (MPG).

La MPG è una istituzione nazionale per la ricerca e ha 76 istituti distribuiti sopra la Germania, ma anche in Olanda ed Italia (Biblioteca Hertziana a Roma e KHI a Firenze).

Il RZG è equipaggiato con una potenza di supercomputer simile a quella di CINECA. Oltre al supercomputing il RZG offre per tutti gli istituti la possibilità di archivare dati per lungo tempo. Questi archivi si trovano nel AFS.

La cella AFS del RZG si chiama (per motivi storici) "ipp-garching.mpg.de".

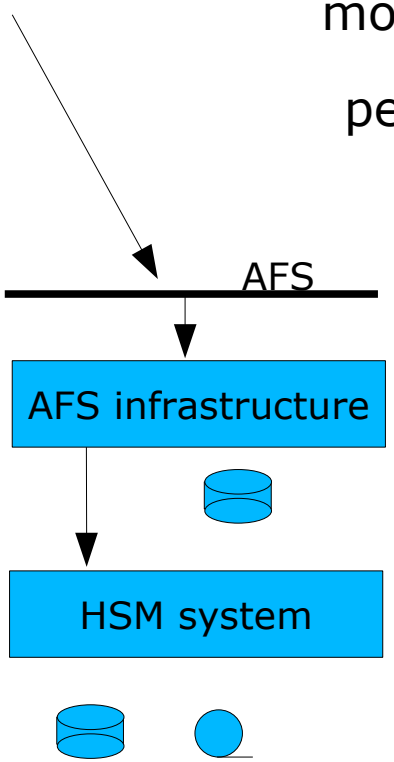
Questa cella contiene una grande quantità di dati provenienti da esperimenti, dalla digitalizzazione di fototeche e da altre fonti.

Siamo obbligati a conservare questi dati per almeno i prossimi 50 anni.

Così la sicurezza dei dati è molto importante.

- Ogni *file* ha al meno una seconda copia sia su disco o su nastro
- *File* nei OSD tipicamente hanno 4 copie su nastro

AFS OSD al RZG



AFS OSD come anche prima MR-AFS sono molto adatti per la conservazione a lungo tempo

perché l'architettura di AFS permette lo spostamento dei dati in modo trasparente al utente

perché si può sostituire in modo trasparente non solo l'hardware, ma anche il sistema HSM

Sostituire il sistema HSM può durare anni perché tutti dati devono essere migrati dai nastri del vecchio ai nastri del nuovo sistema!

L'abbiamo fatto al RZG già due volte

Da DMF di Cray a DMF di SGI (1998)

Da DMF di SGI a TSM-HSM (2006-2008)

Incominceremo quest'anno con lo spostamento da TSM-HSM a HPSS

Cella „ipp-garching.mpg.de“

42 *fileserver* con 215 TB dischi in tre città della Germania

23 *non-archival* OSD con 134 TB dischi

2 *archival* OSD con sistema TSM-HSM usando nastri LTO4

(TSM-HSM viene sostituito da HPSS alla fine del anno)

27000 volumi

8700 utenti

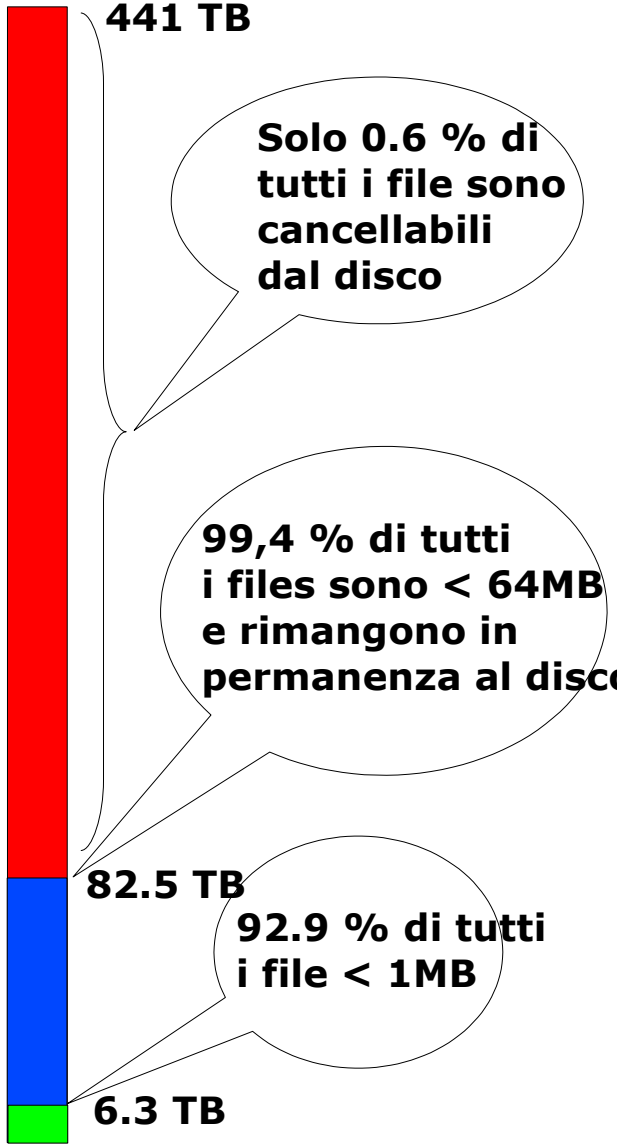
440 TB dati in totale

2.5 TB dati scritti al giorno

3.5 TB dati letti al giorno

File Size Histogram

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	75159746	50.83	50.83	93.416 GB	0.02	0.02
4 KB - 8 KB	11884877	8.04	58.87	65.152 GB	0.01	0.04
8 KB - 16 KB	10260774	6.94	65.81	109.881 GB	0.02	0.06
16 KB - 32 KB	10729387	7.26	73.07	224.105 GB	0.05	0.11
32 KB - 64 KB	9080451	6.14	79.21	413.301 GB	0.09	0.20
64 KB - 128 KB	6951641	4.70	83.91	605.156 GB	0.13	0.33
128 KB - 256 KB	4829427	3.27	87.18	847.677 GB	0.19	0.52
256 KB - 512 KB	4907895	3.32	90.50	1.656 TB	0.38	0.90
512 KB - 1 MB	3601029	2.44	92.93	2.329 TB	0.53	1.43
1 MB - 2 MB	2399004	1.62	94.56	3.338 TB	0.76	2.18
2 MB - 4 MB	2321559	1.57	96.13	6.160 TB	1.40	3.58
4 MB - 8 MB	2117146	1.43	97.56	11.259 TB	2.55	6.13
8 MB - 16 MB	1379340	0.93	98.49	15.003 TB	3.40	9.53
16 MB - 32 MB	769824	0.52	99.01	15.707 TB	3.56	13.10
32 MB - 64 MB	541945	0.37	99.38	22.397 TB	5.08	18.17
64 MB - 128 MB	355160	0.24	99.62	29.960 TB	6.79	24.97
128 MB - 256 MB	191509	0.13	99.75	32.704 TB	7.41	32.38
256 MB - 512 MB	181047	0.12	99.87	66.518 TB	15.08	47.46
512 MB - 1 GB	161963	0.11	99.98	107.424 TB	24.35	71.82
1 GB - 2 GB	20063	0.01	99.99	26.833 TB	6.08	77.90
2 GB - 4 GB	3859	0.00	100.00	10.139 TB	2.30	80.20
4 GB - 8 GB	2176	0.00	100.00	13.321 TB	3.02	83.22
8 GB - 16 GB	1184	0.00	100.00	11.874 TB	2.69	85.91
16 GB - 32 GB	856	0.00	100.00	19.029 TB	4.31	90.23
32 GB - 64 GB	489	0.00	100.00	21.798 TB	4.94	95.17
64 GB - 128 GB	155	0.00	100.00	13.062 TB	2.96	98.13
128 GB - 256 GB	45	0.00	100.00	7.601 TB	1.72	99.85
256 GB - 512 GB	2	0.00	100.00	667.311 GB	0.15	100.00
Totals:	147852553 Files			441.083 TB		

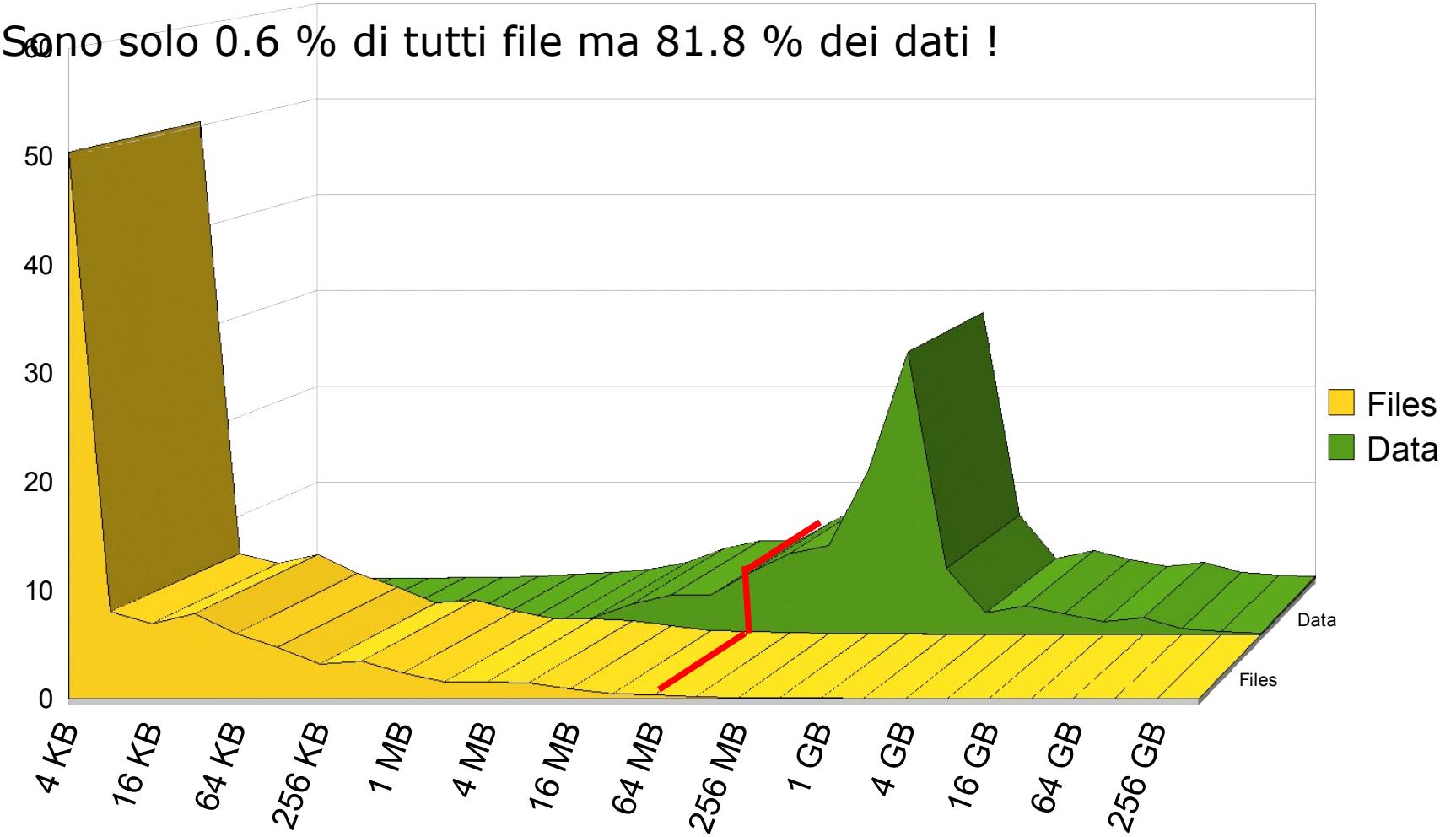


File Size Histogram

Diagramma del numero dei file e del volume di dati in funzione della dimensione.

Solo i file alla destra della linea rossa possono essere cancellati dal disco ed archiviati

Sono solo 0.6 % di tutti file ma 81.8 % dei dati !



Integrazione di AFS OSD

AFS OSD è basato alla versione stabile di OpenAFS (oggi 1.4.11).

E' disponibile sul server *subversion* di DESY. Lo potete "*check out* " con

```
svn co https://svnsrv.desy.de/public/openafs-osd/trunk/openafs
```

La integrazione nel sorgente ufficiale di OpenAFS incomincerà subito dopo la emissione del *release* 1.6 stabile di OpenAFS (quest'anno).

AFS OSD è previsto di far parte del release stabile 1.8 di OpenAFS.

Da parte di RZG, il mio successore Christof Hanke ed io lavoriamo alla integrazione che anche esige un ampliamento dello *standard* AFS3.

Conclusione

In caso che la integrazione di GPFS in AFS OSD vada in produzione a Portici:

Per gli utenti di CRESCO cambierà che

invece di usare direttamente il GPFS possono andare nel albero di AFS dove è montato il suo volume speciale per l'uso di GPFS

hanno bisogno di un *token* AFS anche nei *batch job.*

I file creati in questo modo sono visibili dappertutto, apparendo però sempre in AFS:

Sulle macchine che vedono i file tramite AFS non si vede lo stato attuale di *file* aperti per scrivere.

Per esempio lo stato dei *log-file* si vede completamente solo dopo che il *job* ha finito.

Del resto questo comportamento è quello di AFS che si comporta come d'abitudine.

Domande o Commenti?

Grazie!