Lustre/ZFS Development

Zeuthen Spring 2016

Walter Schoen, GSI

- ZFS at GSI
- ZFS Vectorization (G.Neskovic, Isdma)
- Hidden Markov/lustre logs (T.Stibor, Isdma)
- TSM – Lustre copy tool (T.Stibor, Intel@PCC)

ZFS as Backend System for Lustre

Features (some :-))

- 128 bit COW system (Yotta Byte – „..boiling the ocean ….“)
- data integrity & error correction
  => protection against silent data corruption
- data integrity check/repair on mounted volume with „scrub“
- only „used“ parts of the disks will be rebuild
- pool can be extended „step by step“ with larger disks
- snapshots

GSI:

- used in cluster Nyx as backend FS for the OSTs
- no hardware RAID controllers are used
- due to perfomance issues, still ldiskfs on the MDTs
- RAIDZ2
- in production for >1 year

Experience@GSI:

- reliable & stable +
- no automatic „handling“ of global spares/resilvering -
- => need for „scipting“ automatic resilvering of spare disks

LSDMA project University Frankfurt/GSI (HPC)

„Vectorized ZFS RAIDZ Implementation“: **Gvoszden Neskovic**

Large-Scale-Data-Management -> link: helmholz-lsdma.de

=> idea: vectorizing Galois Field multiplication using modern CPU instruction sets

Error Correction in Zpools

RAIDZ2/3 based on Reed-Solomon-Codes:

- A „signal“ of k numbers should be transmitted without errors
- To add redundancy , the signal is coded as values of a polynom
- Polynoms can be expressed as sum of k monoms
- The coefficients of the monoms are a solution of a linear equation
- Polynom will be extrapolated on n>k grid points
- If m numbers will get lost/damaged, the signal can be reconstructed, if n-m>k
- If two disks get lost, two „syndroms“ P, Q need to be calculated

=>Algebra of a Galois Field

- P is Parity, Q Reed-Solomon-Code

=> lots of multiplications necessary : CPU, perfomance

=> optimisation possible?

Optimisation possible?

=> Vectorisation of the Galois filed multiplicatios
(Gvoszden Neskovec)

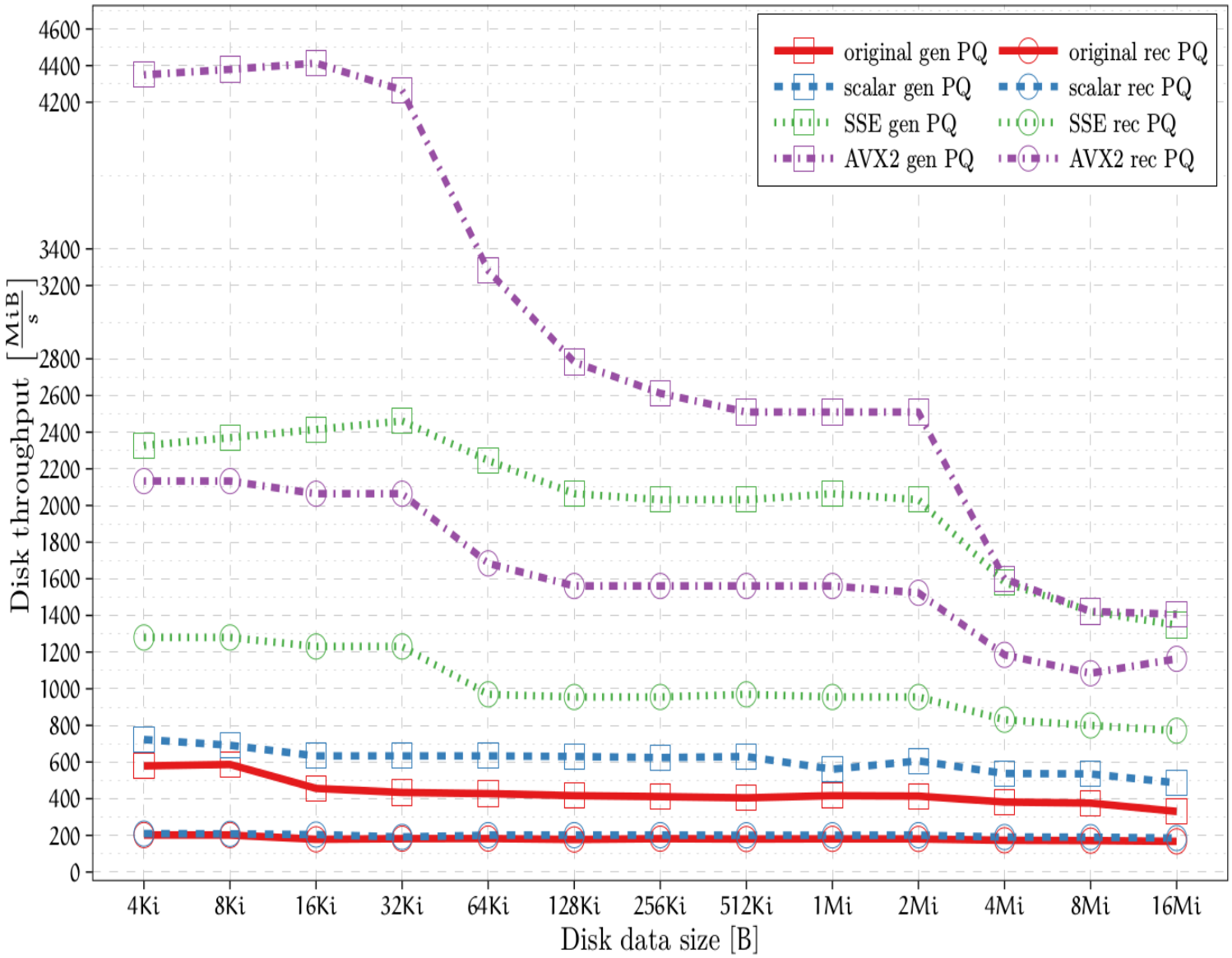
Two Variants:

- SSE variant computes 16 multiplications in parallel
- AVX2 variant computes 32 multiplications in parallel

=> Results

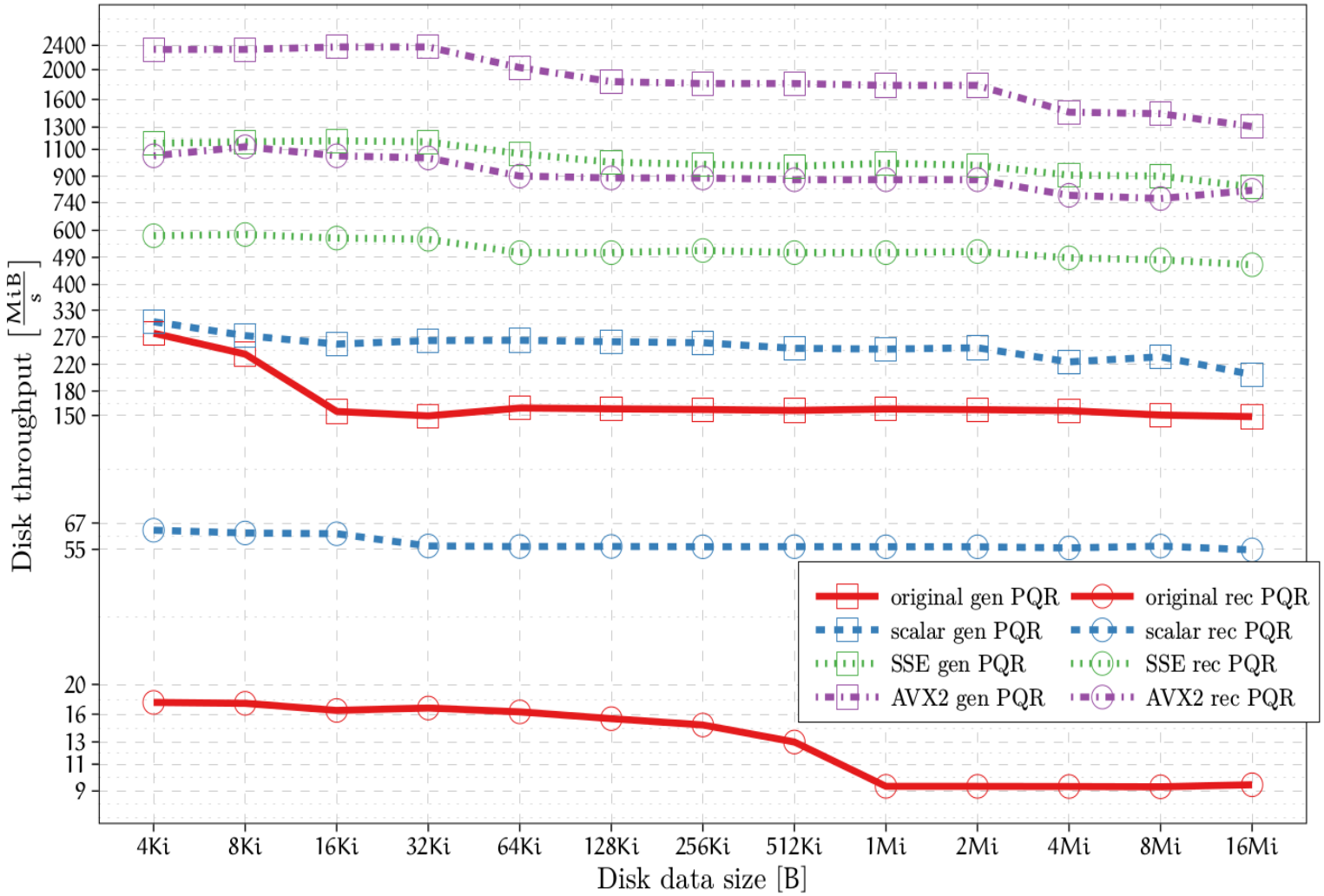
ZFS RAIDZ-2 Results

- RAIDZ-2
- 8 data disks
- 2 parity disks
- Generate PQ
- Reconstruct 2 disks



ZFS RAIDZ-3 Results

- RAIDZ-3
- 8 data disks
- 3 parity disks
- Generate PQR
- Reconstruct 3 disks



Analysing Lustre Log File with a Hidden Markov Model

Is it possible to make predictions of lustre system behavior, based on log information?

E.G predict a „LBUG“ based on the pattern of system calls?

=> Hidden Markov Model : Investigation from **Thomas Stibor**
Screenshots from LUG 14 and lsdma Konferenz at GSI 2016

LBUG's and Implications

- Lustre is a large project with complex code base.
- Lustre is prone to critical software bugs called (LBUG's).
- LBUG is a software behavior that causes freeze of kernel thread and subsequent reboot.

```
void lbug_with_loc(struct libcfs_debug_msg_data *) __attribute__((noreturn));

#define LBUG()                                                          \
do {                                                                    \
        LIBCFS_DEBUG_MSG_DATA_DECL(msgdata, D_EMERG, NULL);             \
        lbug_with_loc(&msgdata);                                        \
} while(0)

#define LIBCFS_DEBUG_MSG_DATA_DECL(dataname, mask, cdls)     \
        static struct libcfs_debug_msg_data dataname = {     \
                .msg_subsys = DEBUG_SUBSYSTEM,               \
                .msg_file   = __FILE__,                      \
                .msg_fn     = __FUNCTION__,                  \
                .msg_line   = __LINE__,                      \
                .msg_cdls   = (cdls)          };             \
        dataname.msg_mask   = (mask);
```

"Note - LBUG freezes the thread to allow capture of the panic stack. A system reboot is needed to clear the thread."

From: http://wiki.lustre.org/manual/LustreManual20_HTML/LustreTroubleshooting.html

Lustre Log Data Pre-Processing Steps

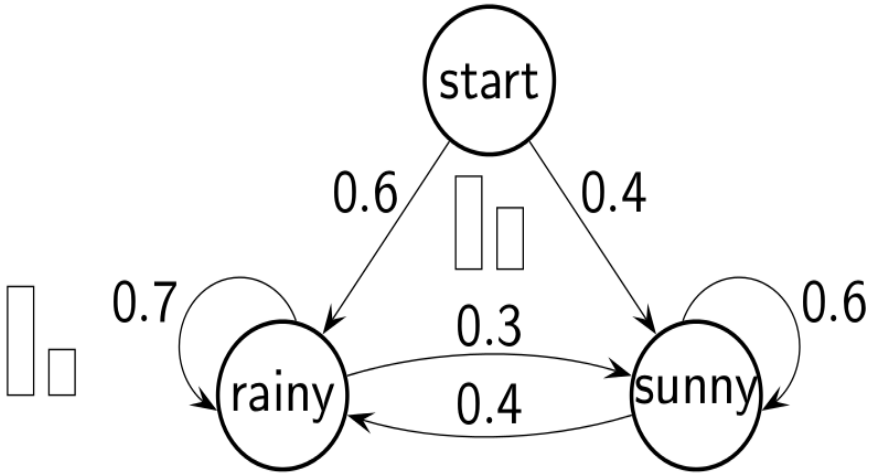
1. Fetching entire log data from archive tape.
2. Resulting in "giant" log data.

```
drwxr-sr-x 14 root staff   114 Mai   2 11:37 ..
-rw-r-----  1 root adm    2,5G Feb   2  2012 syslog-20120201
-rw-r-----  1 root adm    2,0G Feb   3  2012 syslog-20120202
-rw-r-----  1 root adm    2,2G Feb   4  2012 syslog-20120203
-rw-r-----  1 root adm    2,0G Feb   5  2012 syslog-20120204
-rw-r-----  1 root adm    1,9G Feb   6  2012 syslog-20120205
-rw-r-----  1 root adm    1,9G Feb   7  2012 syslog-20120206
(...)

du -sh 2012/02/
57G     02/
```

- Total amount of Lustre log data for 2012 is $\approx$ 2.1 GByte.
- A true Big Data problem.

Markov Model Weather Example



$$\mathbf{s} = \begin{array}{c} \begin{array}{cc} \text{rainy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.6 & 0.4 \end{pmatrix} \end{array}$$

$$\mathbf{H} = \begin{array}{c} \\ \text{rainy} \\ \text{sunny} \end{array} \begin{array}{c} \begin{array}{cc} \text{rainy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \end{array}$$

T	:	Length of observation sequence.
N_Q	:	Number of states in the model.
$\mathbf{s}$	:	Initial state distribution.
$\mathbf{H}$	:	Transition matrix.
$\{Q_1, Q_2, \ldots, Q_T\}$	:	Set of time indexed random variables for states.
$\mathfrak{M} = (\mathbf{s}, \mathbf{H})$	:	Markov model parameters.

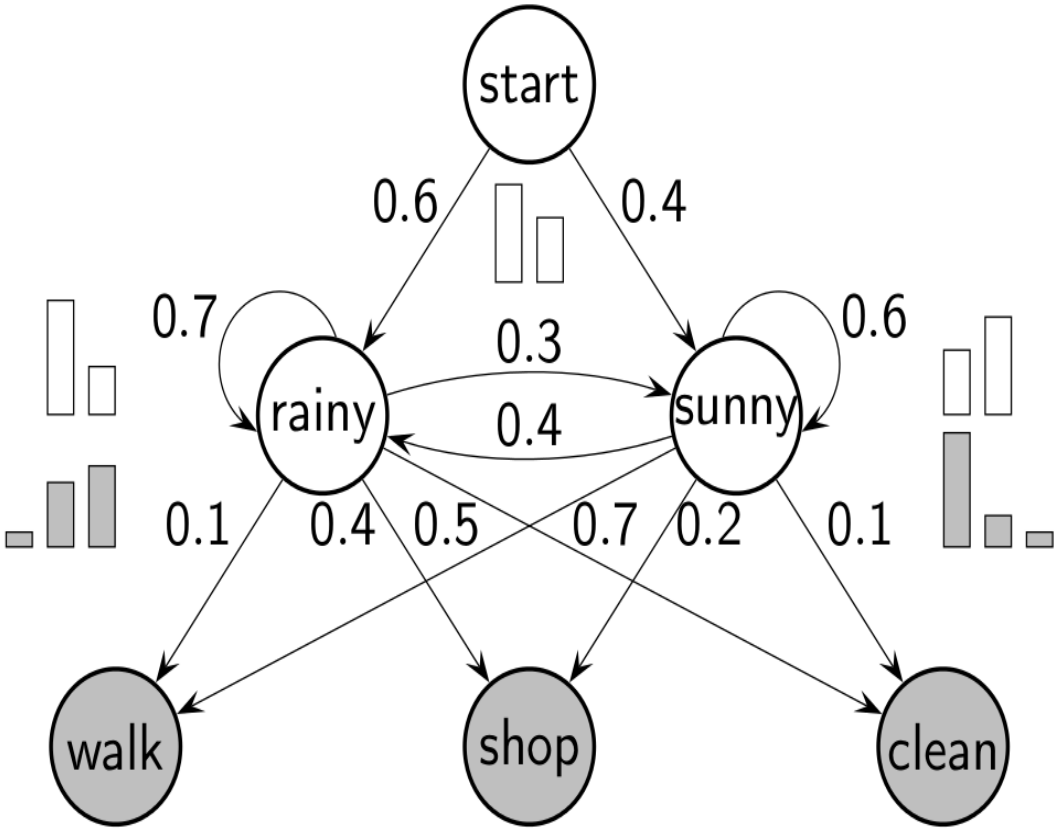
Joint distribution for *sequence* of T observations
$\Pr(Q_1, Q_2, \ldots, Q_T) = \Pr(Q_1) \prod_{t=2}^{T} \Pr(Q_t | Q_1, \ldots, Q_{t-1})$

Joint distribution under *first-order* Markov assumption:
$\Pr(Q_1, Q_2, \ldots, Q_T) = \Pr(Q_1) \prod_{t=2}^{T} \Pr(Q_t | Q_{t-1})$

Hidden Markov models

Suppose we are locked in room without windows, and somebody is telling us the following observations and ask us to tell him what weather is outside:

$$\mathcal{O} = \text{walk} \rightarrow \text{clean} \rightarrow \text{shop} \rightarrow \cdots \rightarrow \text{shop}$$

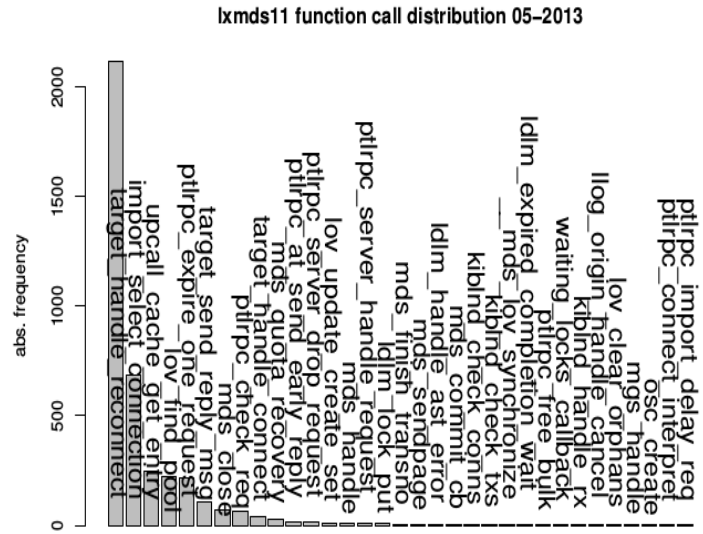


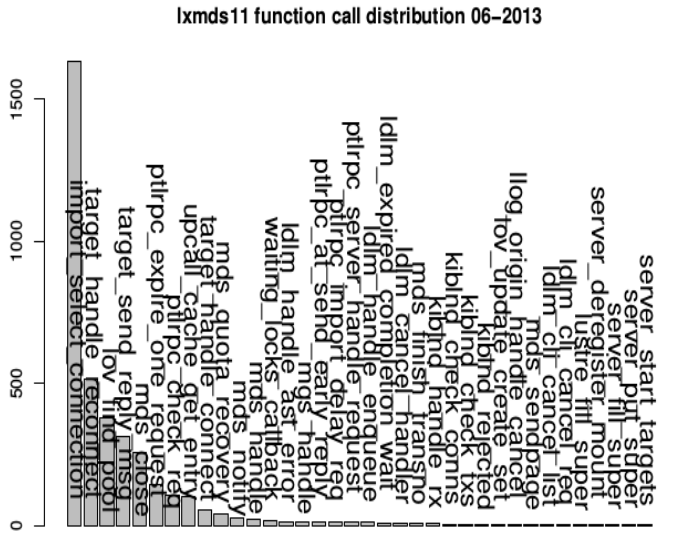
$$\mathbf{s} = \begin{matrix} \text{rainy} & \text{sunny} \end{matrix} \begin{pmatrix} 0.6 & 0.4 \end{pmatrix}$$

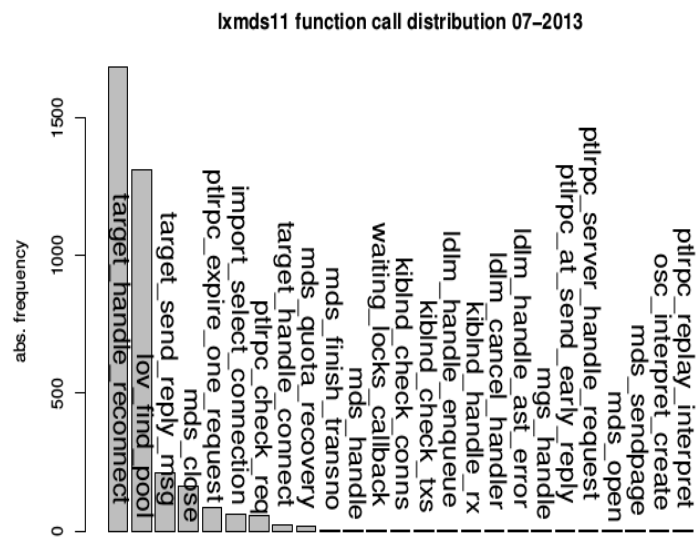
$$\mathbf{H} = \begin{matrix} & \begin{matrix} \text{rainy} & \text{sunny} \end{matrix} \\ \begin{matrix} \text{rainy} \\ \text{sunny} \end{matrix} & \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \end{matrix}$$

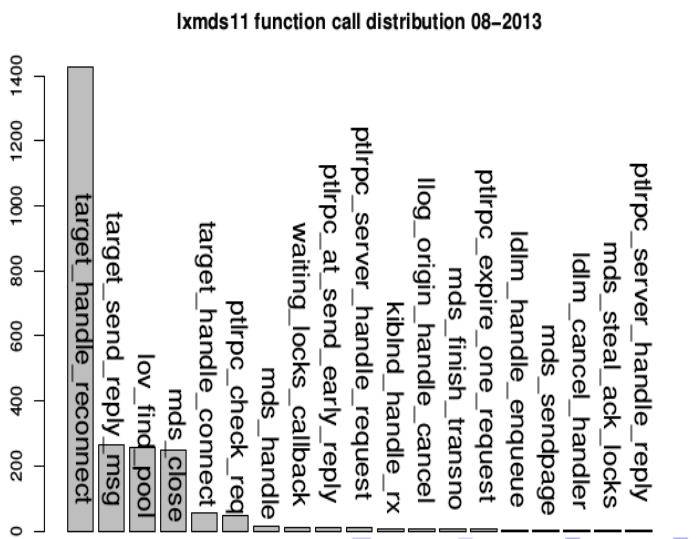
$$\mathbf{E} = \begin{matrix} & \begin{matrix} \text{walk} & \text{shop} & \text{clean} \end{matrix} \\ \begin{matrix} \text{rainy} \\ \text{sunny} \end{matrix} & \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{pmatrix} \end{matrix}$$

Frequency Distribution of Func. Calls (1-Gram Seq.) (cont.)

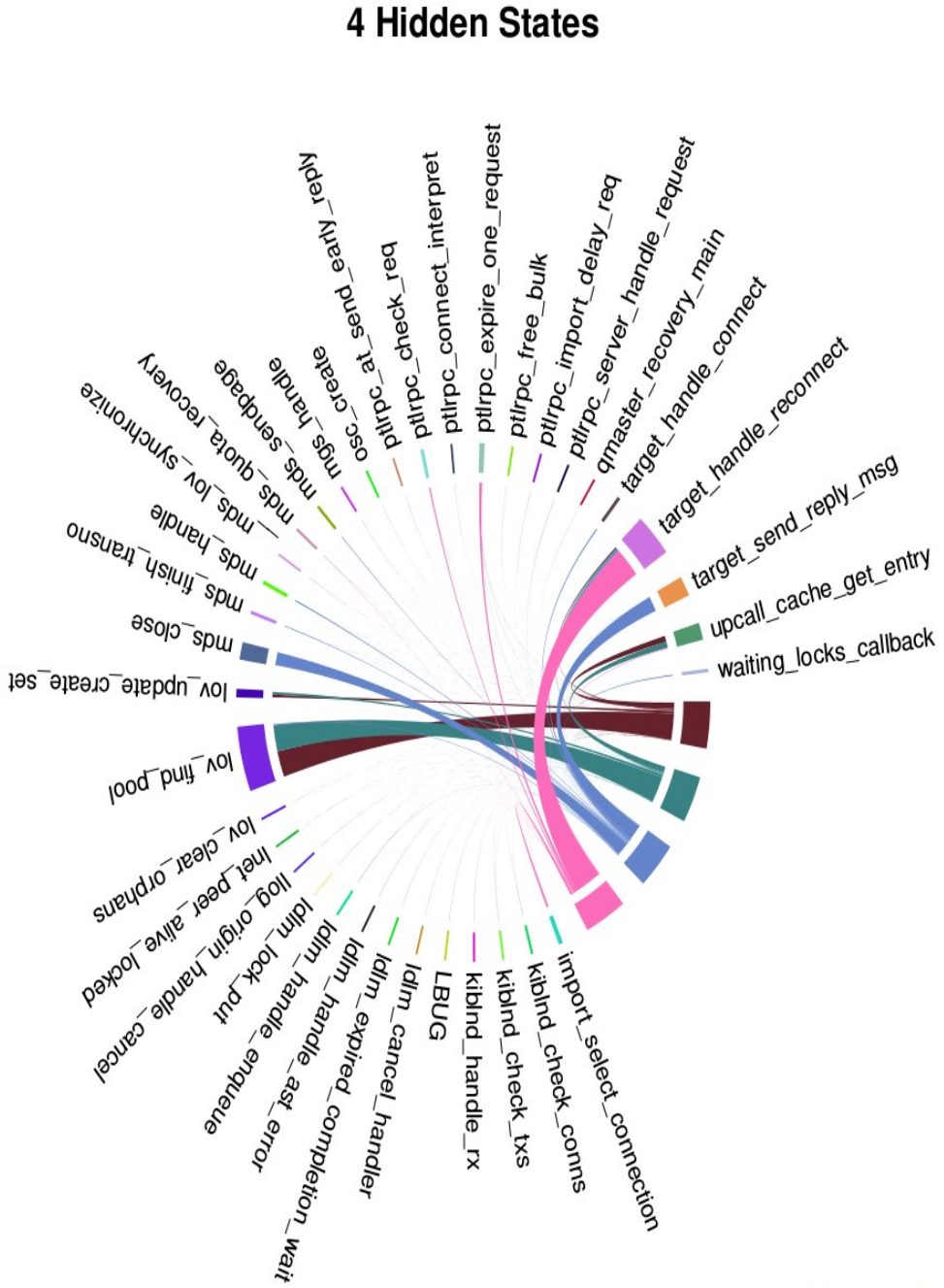








Visualize HMM of Functions Calls (cont.)



GSI is now a Intel Parallel Computing Center

Link: software.intel.com/de-de/ipcc

Project: Developping a Lustre TSM copy tool

- TSM (tivoli storage manager) is a very powerfull tape archive/backup system
- Lustre is a very powerfull parallel file system :-)
- Lustre includes HSM features

=> Goal is: To combine this systems in an efficient (parallel) approach
to have a Lustre/HSM system with a TSM servers + tape robot as backend

Lustre TSM CopyTool

GSI Project:

Developing robust TSM CopyTool for TSM storage backends which seamless integrates into Lustre HSM framework.

Advanced Lustre* Research
Intel Parallel Computing Centers

Uni Hamburg + German Client Research Centre (DKRZ)
- Adaptive optimized ZFS data compression
- Client-side data compression

GSI Helmholtz Centre for Heavy Ion Research
- TSM HSM copytool for Lustre

University of California Santa Cruz
- Automated client-side load balancing

Johannes Gutenberg University Mainz
- Global adaptive Network Request Scheduler

Lawrence Berkeley National Laboratory
- Spark and Hadoop on Lustre

TSM Overview

Tivoli Storage Manager is a client/server product for backup and archive in heterogeneous distributed environments.

Storage hierarchies: Automatically move data from one device to another (or one media type to another) based on characteristics such as file size or storage capacity.

Data collocation: Storing client or group of clients in few number of tapes as much as possible, this process is called collocation.

Deduplication: Eliminating duplicate copies of repeating data.

Compression: Compress data stream seamless either on client or server side.

Image from Wikipedia

Lustre TSM CopyTool (Phase 1, TSMAPI)

```
object # 1
fs: /, hl: /archives/imgs, ll: /a.jpg
object id (hi,lo)                          : (0,74330)
object info length                         : 32
object info                                : inode:17704735,uid:1000,gid:1000
archive description                        : md5sum 45
     a9456eba82b9ee031b3c88fbdd99f9
owner                                      : tstibor
insert date                                : 2016/3/22 14:38:48
expiration date                            : 2017/3/22 14:38:48
restore order (top,hi_hi,hi_lo,lo_hi,lo_lo): (10,0,17583,0,0)
estimated size (hi,lo)                     : (0,7930)
object # 2
fs: /, hl: /archives/imgs, ll: /b.jpg
object id (hi,lo)                          : (0,74331)
object info length                         : 32
object info                                : inode:17704738,uid:1000,gid:1000
archive description                        : Lustre fid [0x5100000dd9:0xeb:0x0]
owner                                      : tstibor
insert date                                : 2016/3/22 14:38:48
expiration date                            : 2017/3/22 14:38:48
restore order (top,hi_hi,hi_lo,lo_hi,lo_lo): (10,0,17584,0,0)
estimated size (hi,lo)                     : (0,8358)
```

Lustre related HSM meta information can be stored in char arrays object info and description each of length 256 bytes. In this example inode,uid,gid,fid and md5sum information are stored.

Initial `tsmapi.h` (later linked with Lustre HSM framework)

```
#include "dsmapifp.h"
#include "dsmapips.h"
#include "dsmrc.h"
#include "dapint64.h"
#include "log.h"

#define TSM_BUF_LENGTH 65536
#define MAX_PASSWORD_LENGTH 32
#define MAX_USERNAME_LENGTH 32
#define MAX_OFF_T_DS64 20

#define MAGIC_ID 71147

typedef struct {
    char node[DSM_MAX_ID_LENGTH + 1];
    char password[MAX_PASSWORD_LENGTH + 1];
    char username[MAX_USERNAME_LENGTH + 1];
    char platform[DSM_MAX_PLATFORM_LENGTH + 1];
} login_t;

typedef struct {
    qryRespArchiveData qry_resp_arv_data;
} query_node_t;

typedef struct {
    unsigned long capacity;     /* Number of maximum elements. If max capacity is reached,
                                   then capacity is doubled  by means realloc. */
    unsigned long N;            /* Number of actual elements. */
    query_node_t **query_node;
} query_arr_t;

typedef struct {
    dsUint64_t f_seq;   /* __u64 */
    dsUint32_t f_oid;   /* __u32 */
    dsUint32_t f_ver;   /* __u32 */
} lu_fid_t;

typedef struct {
    unsigned int magic;
    dsUint8_t objType;
    dsStruct64_t size;
    lu_fid_t lu_fid;
 } obj_info_t;

off_t to_off_t(dsStruct64_t *size); /* Convert dsStruct64_t to off_t required for filesize conversation. */
dsStruct64_t to_dsStruct64_t(const off_t size); /* Convert off_t to dsStruct64. */

dsInt16_t tsm_init(login_t *login);
void tsm_quit();

dsInt16_t tsm_query_session_info();
dsInt16_t tsm_archive_file(const char *fs, const char *filename, char *desc);
dsInt16_t tsm_query_hl_ll(const char *fs, const char *hl, const char *ll, dsBool_t display);
dsInt16_t tsm_query_file(const char *fs, const char *filename, dsBool_t display);
dsInt16_t tsm_delete_file(const char *fs, const char *filename);
dsInt16_t tsm_delete_hl_ll(const char *fs, const char *hl, const char *ll);
dsInt16_t tsm_retrieve_file(const char *fs, const char *filename);
dsInt16_t tsm_retrieve_hl_ll(const char *fs, const char *hl, const char *ll);

#endif /* TSMAPI_H */
```

Lustre TSM CopyTool (Phase 1, TSMAPI) (cont.)

```
[DEBUG] (src/tsmapi.c) (dsmBeginQuery:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmGetNextQObj:handle:1:ANS0258I (RC2200) On dsmGetNextQObj or dsmGetData there is more available data)
[DEBUG] (src/tsmapi.c) (dsmGetNextQObj:handle:1:ANS0272I (RC121)  The operation is finished)
[DEBUG] (src/tsmapi.c) (dsmBeginTxn:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmDeleteObj:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmEndTxn:handle:1:ANS0302I (RC0)    Successfully done.)
[VERBOSE]
Deleted obj fs: /
        hl: /tmp/ttsm/archives/linux/fs
        ll: /pipe.c
+ set +x
+ bin/ltsm --delete -f / --node lxdv81 --password lxdv81 /tmp/ttsm/archives/linux/fs/file_table.c
[VERBOSE]
option -d, --delete
[VERBOSE]
option -f, --fsname=/
[VERBOSE]
option -n, --node=lxdv81
[VERBOSE]
option -p, --password=lxdv81
[DEBUG] (src/tsmapi.c) (dsmInitEx:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmRegisterFS:handle:1:ANS0242W (RC2062) On dsmRegisterFS the filespace is already registered)
[DEBUG] (src/tsmapi.c) (dsmQuerySessOptions:handle:1:ANS0302I (RC0)    Successfully done.)
[VERBOSE]
DSMI_DIR        : /opt/tivoli/tsm/client/api/bin64
DSMI_CONFIG     : /tmp/ttsm/dsmopt/dsm.opt
serverName      : LXDV81-KVM-TSM-SERVER
commMethod      : 1
serverAddress   : 192.168.254.101
nodeName        : LXDV81
compress        : 0
compressalways  : 1
passwordAccess  : 0
[DEBUG] (src/tsmapi.c) (dsmQuerySessInfo:handle:1:ANS0302I (RC0)    Successfully done.)
[VERBOSE]
Server's ver.rel.lev       : 7.1.3.0
ArchiveRetentionProtection : No
[VERBOSE]
Max number of multiple objects per transaction: 4096
Max number of Bytes per transaction: 26214400
dsmSessInfo.fsdelim: /
dsmSessInfo.hldelim: /
[VERBOSE]
API Library Version = 7.1.3.0
[VERBOSE]
tsm_query_archive with settings
fs: /
hl: /tmp/ttsm/archives/linux/fs
ll: /file_table.c
owner:
descr: *
[DEBUG] (src/tsmapi.c) (dsmBeginQuery:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmGetNextQObj:handle:1:ANS0258I (RC2200) On dsmGetNextQObj or dsmGetData there is more available data)
[DEBUG] (src/tsmapi.c) (dsmGetNextQObj:handle:1:ANS0272I (RC121)  The operation is finished)
[DEBUG] (src/tsmapi.c) (dsmBeginTxn:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmDeleteObj:handle:1:ANS0302I (RC0)    Successfully done.)
[DEBUG] (src/tsmapi.c) (dsmEndTxn:handle:1:ANS0302I (RC0)    Successfully done.)
[VERBOSE]
Deleted obj fs: /
        hl: /tmp/ttsm/archives/linux/fs
        ll: /file_table.c
+ set +x
##### Deleting done #####
Successfully archived and retrieved data verified with MD5SUM
```

Raw “-lApiTSM64” calls

Example output of developed simple console client (called `ltsm`) using `tsmapi.h`

```
tstibor@lxdv81:/tmp/ltsm>./bin/ltsm
missing or wrong argument combinations of archive, retrieve, query or delete
syntax: ./bin/ltsm
        -a, --archive
        -r, --retrieve
        -q, --query
        -d, --delete
        -f, --fsname <STRING>
        -h, --hl <STRING>
        -l, --ll <STRING>
        -n, --node <STRING>
        -u, --username <STRING>
        -p, --password <STRING>
tstibor@lxdv81:[255]/tmp/ltsm>
```

Lustre TSM CopyTool (Phase 2 and beyond)

- Integrating `tsmapi.h` into Lustre HSM framework
- Extending `tsmapi.h` with new compression stream algorithm such as LZ4 (perform a comparison to TSM built-in compression)
- Exploit all optimizations provided by TSM such as
  - Maxinum number of multiple objects per transaction
  - Efficient data structures for storing querying data

Will be soon available at github.

Thanks to: G.Neskovic, T.Stibor

Walter Schön, GSI