Summary of 1 TB Milestone

RD Schaffer

Outline:

- Goals and assumptions of the exercise
- The hardware constraints
- The basic model
- What have we understood so far
- Where do we go from here
- Summary

Thanks to those who contributed to the 1 TB test

I would like to thank those people who contributed to the successful completion of the 1 TB milestone:

Martin Schaller and Rui Silva Carapinha from Atlas

Gordon Lee, Alessandro Miotto, and Harry Renshall from the IT/PDP group

Dirk Duellmann and Marcin Nowak from IT/ASD group

Basic goals of the 1 TB test

The primary goals:

- Write 1 TB of simulated raw data (***jet production digits***) to Objy databases stored in HPSS
  - ➨ Demonstrate the feasibility of the different elements with a first approximation of a model for Atlas raw data
  - ➨ Understand the performance of the different elements:
    - basic hardware configuration, raw data object model

Learning from this:

- Develop a system capable of easily loading Objy databases with Zebra data at a few MB/sec

Globally, what has been achieved

We have written 1 TB of jet production data into Objy databases stored in HPSS

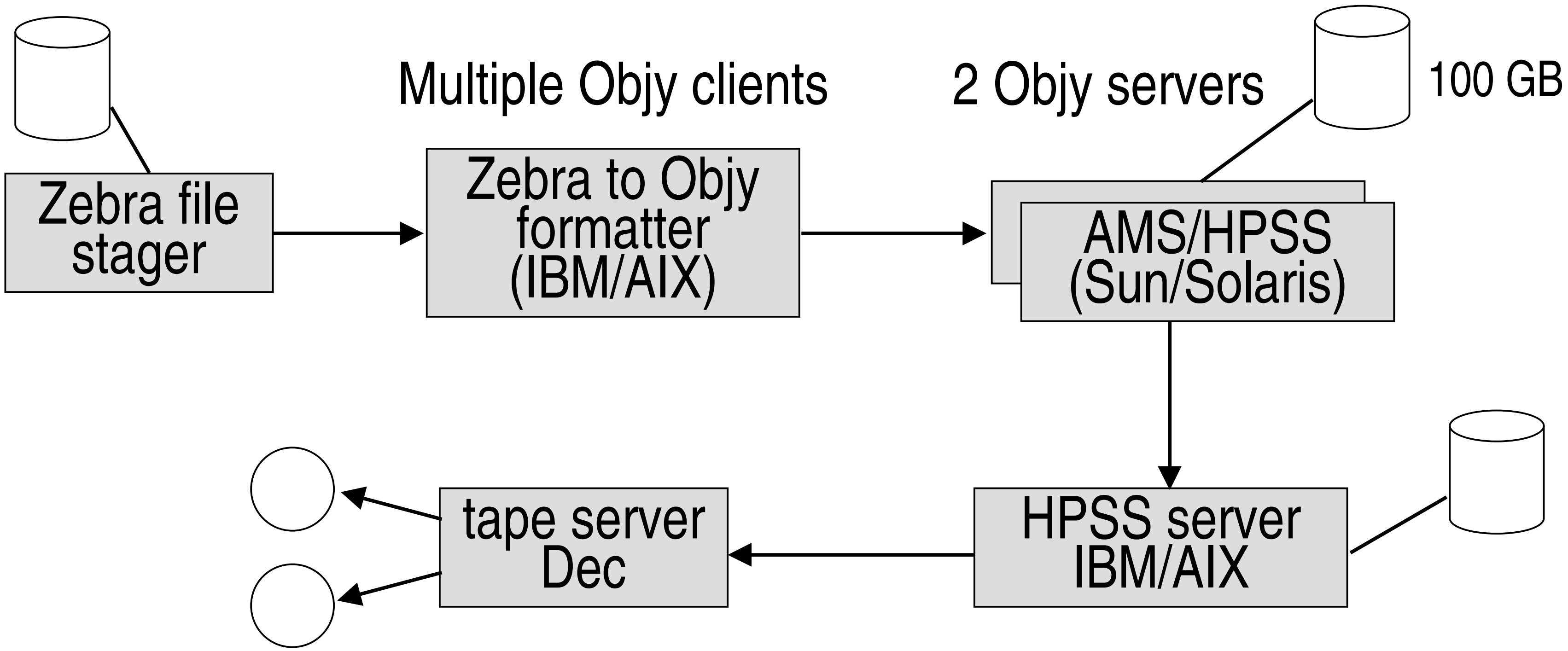
Overall performance of 1 TB:

- 5 ibm/aix Objy clients writing to 2 sun/solaris Objy servers
- typical aggregate write speed:
  - ~1.5 MB/sec ***with*** HPSS staging-out
  - ~3 MB/sec ***without*** HPSS staging-out
- operational efficiency over X-mas break: ~50%
  - ➨ ~19 days to write 1 TB

The observed performance has not yet been fully understood

The hardware configuration

The basic hardware configuration was a client-server model:



The hardware constraints

The hardware constraints:

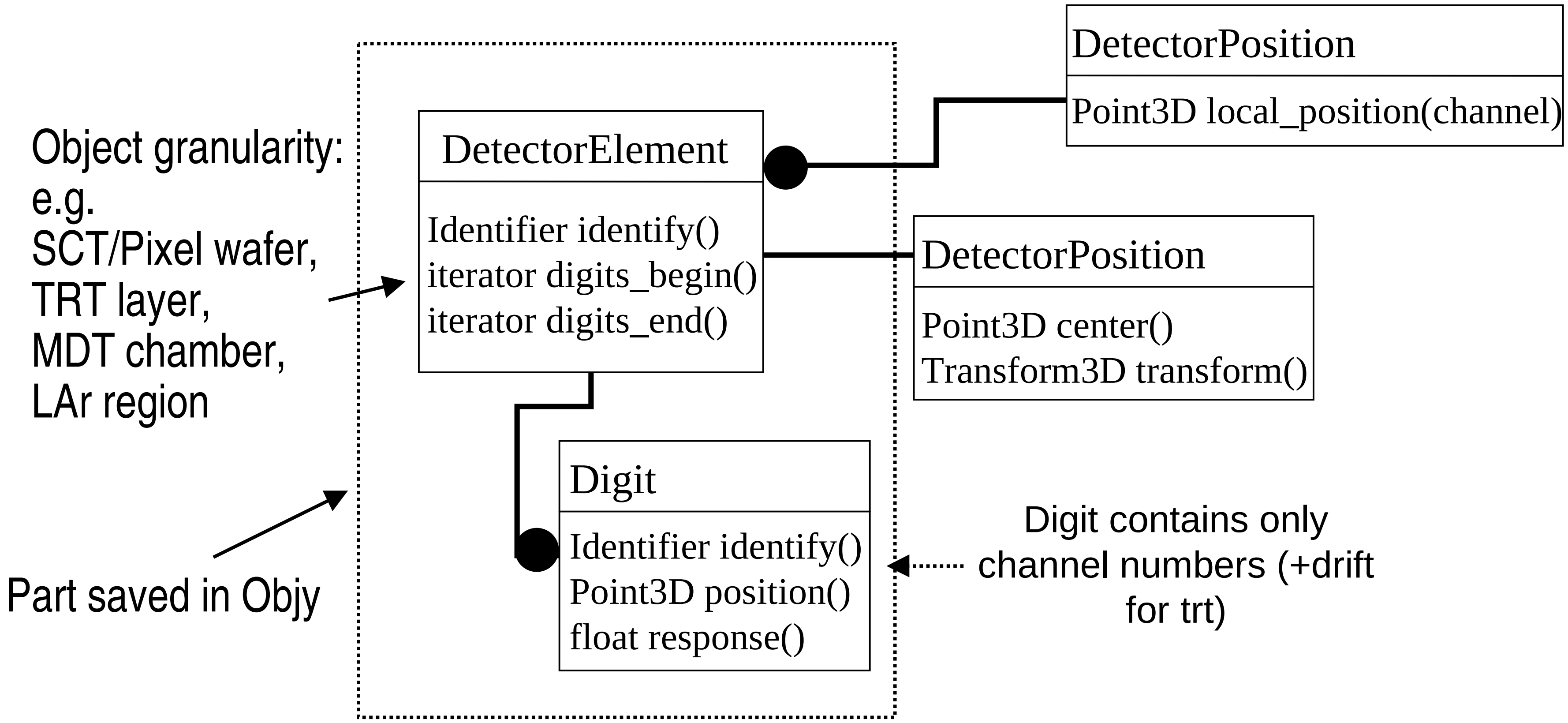
- Limited to clients running Objy V4 on AIX machines
  - Atlas software (Fortran + C++) releases were available only on hp/ibm/dec => hp needed V5.1 and dec needed V5

➨ No tests could be done with client on Sun server to and bypass AMS

➨ Forced dependence on network connection Sun/AIX machines

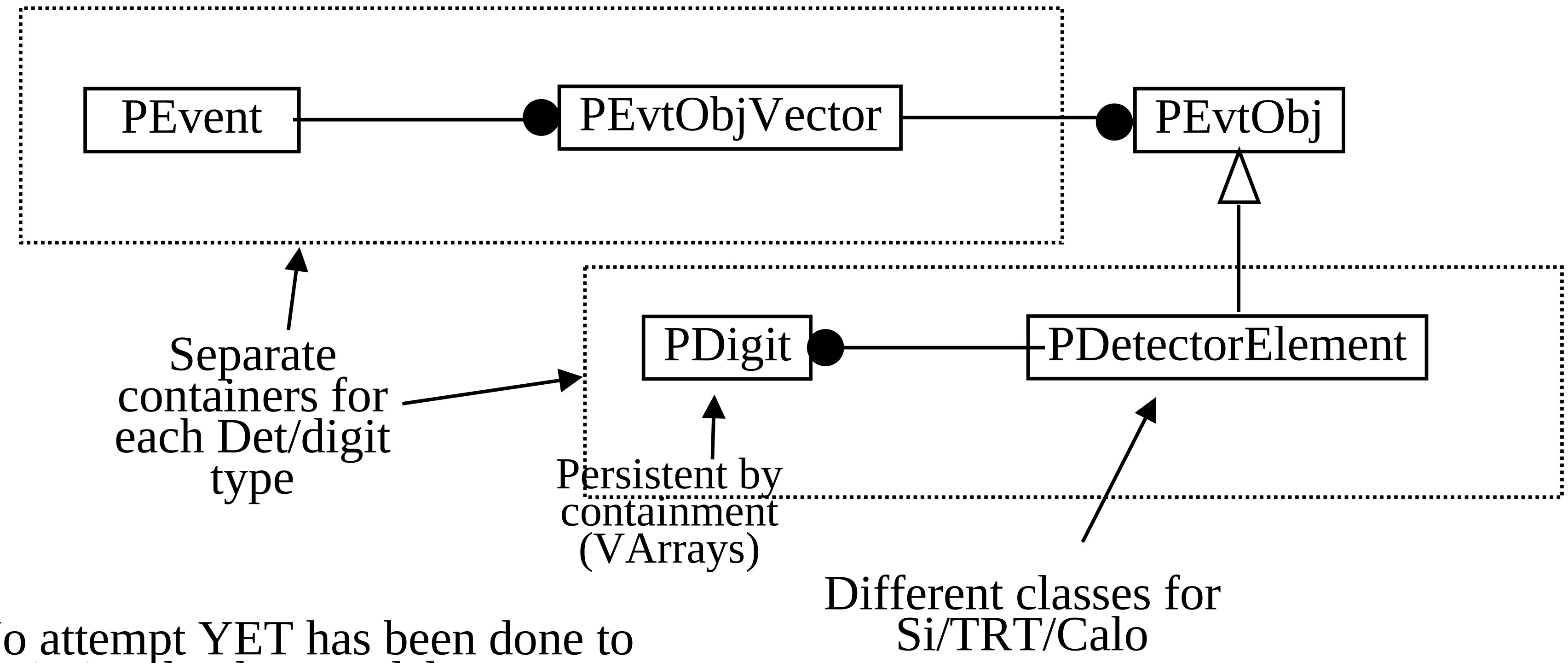
The basic model

Recall the basic raw data transient model:



The basic model, cont.

The basic persistent model is:



No attempt YET has been done to optimize the data model.

The basic model, cont

In order to limit the amount of data and cpu required by the conversion application:

- ➨ digits were duplicated by x10

Typical event size: ~3 MB (jet production, with digit dupl.)

Sizes of VArrays: ~100 B, ~1000 B, ~15 000 B

6%   66%   24% of total data

Space overhead: ~15% (1 - byte count/db size)

(no ootidy run, no data compression tried)

Objy page size was 8 kB

What have we understood so far

What are the different elements which can limit the I/O throughput:

- Objy server <-> disk I/O
- Objy <-> HPSS interface
- Objy client <-> Objy server communication
  - AMS
  - network
- Objy application (Zebra read, conversion, Objy write)

We have been investigating these different elements

What have we understood, cont.

Objy server <-> disk I/O and HPSS interface:

- The 1 TB test was done with a single SCSI disk per server with typical I/O speed of **10 MB/sec**
  - We have seen concurrent read/write reduces throughput by factor of 2
    - ➨HPSS staging causes factor of 2 reduction for SCSI disks.
- Since the 1 TB test, the main Objy server (atlobj02) has been equipped with a RAID disk array:
  - Aggregate I/O rates for multiple stream read/write are **~25 to 30 MB/sec** (simple I/O, I.e. NOT Objy)
  - I/O rates are ~ independent of number of streams

What have we understood, cont.

Objy performance on Objy server, i.e. local disk r/w:

Marcin Nowak has made r/w measurements with a simple application (2 GB db, 2 kB objects)

- write speed 11 - 14 MB/sec (8 kB and 32 kB page sizes)
- read speed ~ 25 MB/sec (8 kB and 32 kB page sizes)

  - ➨ ~x2 loss for local write for this model
  - ➨ confirms other measurements where Objy read speed is ~80% of simple disk read

What have we understood, cont.

Objy performance from remote client (atlobj01/sun), i.e. add AMS and network (**tcp speed ~11 MB/sec**):

Corresponding measurements by Marcin Nowak:

- write speed 6 - 8 MB/sec (8 kB and 32 kB page sizes)
- read speed 3 - 5 MB/sec (8 kB and 32 kB page sizes)

Interaction of the network + AMS clearly:

- ➨reverses the I/O of read and write, and
- ➨introduces an additional throughput loss

The detailed reasons for this remain to be understood

What have we understood, cont.

Network configurations:

Although the two sun servers have a network connection of **~11 MB/sec**,

Other computing platforms, e.g. rsplus/atlaswgs/atlaslinux, have a worse (and unknown) network connectivity:

measurements typically give **3 - 4 MB/sec**

➡ Over the next few weeks, this is a general upgrade of the network connectivity to GB ethernet which should bring connections between atlaswgs and atlobj01/02 to **~10 MB/sec**

What have we understood, cont.

Performance of the Objy application:

Up to now, this has not yet been thoroughly investigated:

- it has been clear that bottlenecks have been elsewhere (at least for events with duplicated digits)
- it will help to have full Atlas software built on sun to understand the local r/w performance

However, there is indication that work needs to be done:

For example, I am now able to read through a database locally on atlobj02 and find **<7 MB/sec** where Marcin Nowak's simple model gives **25 MB/sec** read performance.

Where do we go from here

Clearly the performance issues must be fully understood.

We would like to reach the point where we can repeat the 1 TB milestone with ~5 MB/sec average write speed

- ➨ should take ~3 days

In parallel, we are creating an environment where Objy can be used in a more general way.

Where do we go from here, cont.

Creating a working environment to use Objy:

- We have been porting the Objy software to hp/dec/linux/sun using Objy v5.1
- We currently have three Objy machines:
  - atlobj01 - a developers server, for personal boot files and developer db's, has ~100 GB disk space
  - atlobj02 - production server with RAID disk backed up by HPSS
    - This will soon be replaced by another sun, and atlobj02 will then become an WGS.
  - lockatl - production lock server for production boot/journal files

Where do we go from here, cont.

Creating a working environment to use Objy, cont.:

- For people working on OO reconstruction developments, I would like to see the following scenario:
  - stabilize db schema (e.g. for different production releases)
  - move “standard subsets” of the Geant3 data to atlobj02
  - reconstruction developers then work on atlaswgs accessing these subsets via atlobj02

As the db schema evolves, this cycle will have to be repeated creating new federations for the different production releases.

Summary

As a first exercise, we have been able to write 1 TB of Atlas raw data into Objy db's and HPSS over the X-mas holidays

The average write performance was ~1.5 MB/sec with a duty cycle of ~50%.

Although not fully understood, elements limiting performance have been identified to be: network, disk r/w capabilities, use of AMS, and (possibly) the event model

We hope to repeat this test with ~5 MB/sec capability, and set up a production environment for reconstruction developers to work in.