Overview of pTool and progeny

Stu Fuess

D0 Languages Working Group Meeting

July 19, 1995

pTool and progeny

pTool is a "persistent object manager" originally developed as part of the PASS project

- "persistent", in that is deals with objects with a lifetime beyond program execution: memory/disk/tape
- "object", in that it deals with structures defined in C++ classes
  - a caveat... what is persistent is not really the object, but only the structure. Member functions are not persistent!
- The application programmer is provided with a C++ class with the pTool API.

As part of the CAP project, FNAL modified pTool to make "Fermi-pTool".

A next generation FNAL-only product, POPM, is under construction.

Object Identification

There are two ways to identify an object:

- By an identifier (ID)
  - from which an address can be determined
    - « by lookup, eg a database
    - « by computation
  - Flexible

  Example: file system

- By an address
  - Fast

  Example: pointers

Which method is best for identifying events (or components of events)? Some combination of schemes may be best!

The persistent pointer

The basis of pTool is the use of a 64-bit "persistent pointer" (pptr), which extends the concept of a normal memory pointer to encompass persistent structures.

The components of the pTool pptr are:

- Store number

  Each object is associated with a store

- Folio number

  A folio is equated with a UNIX file, many of which make up a store

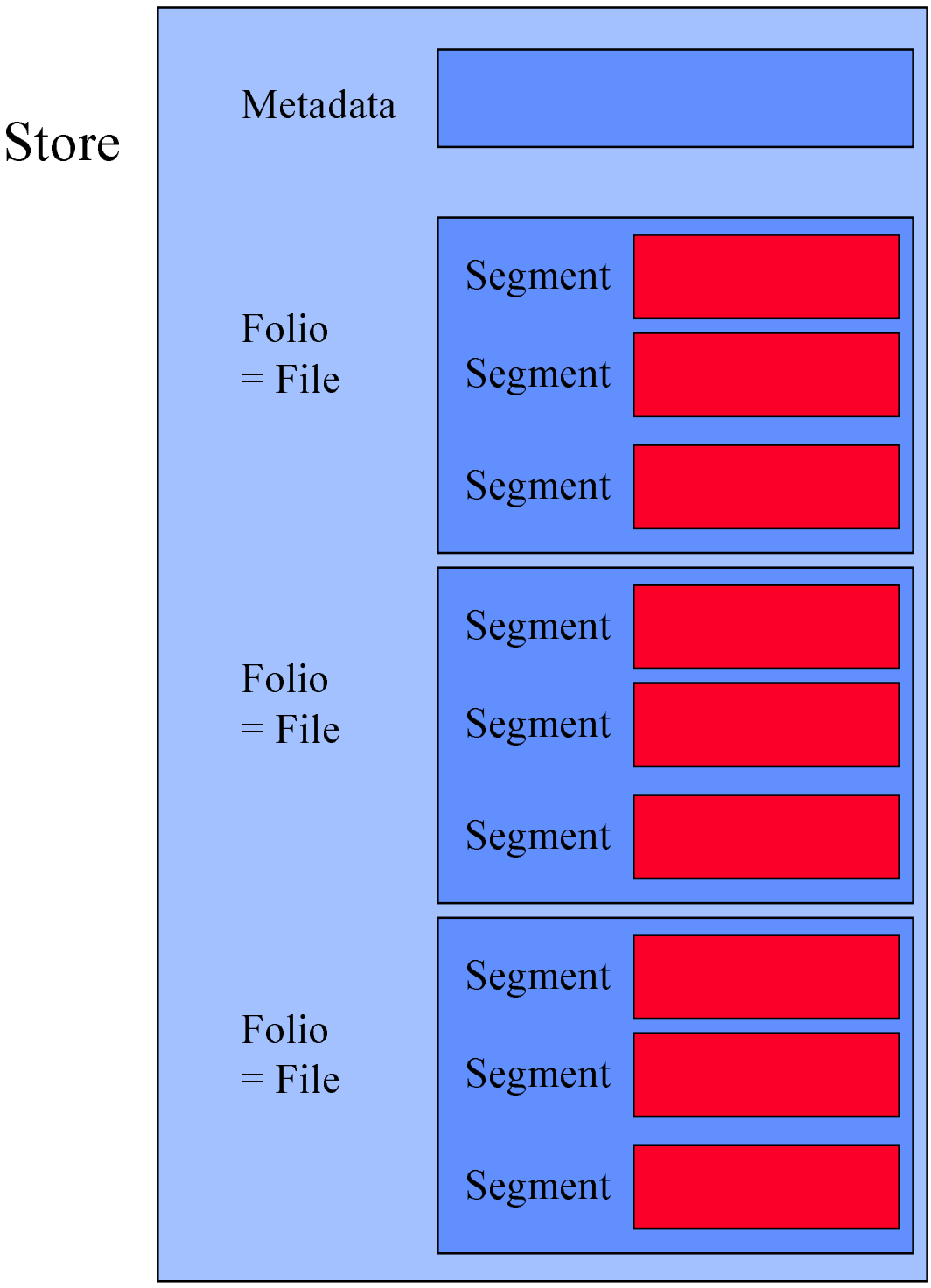
- Segment number

  A folio/file is broken into segments of size appropriate to I/O functions

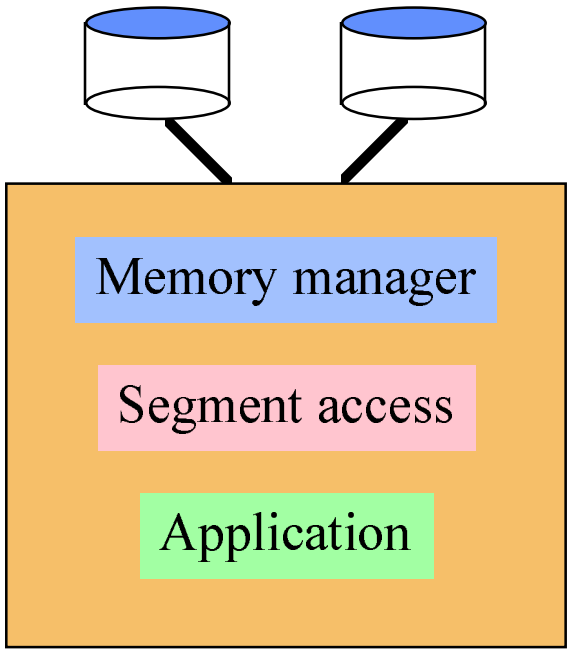
- Byte offset

  Determines the ultimate location of the datum of interest

pTool Data Organization

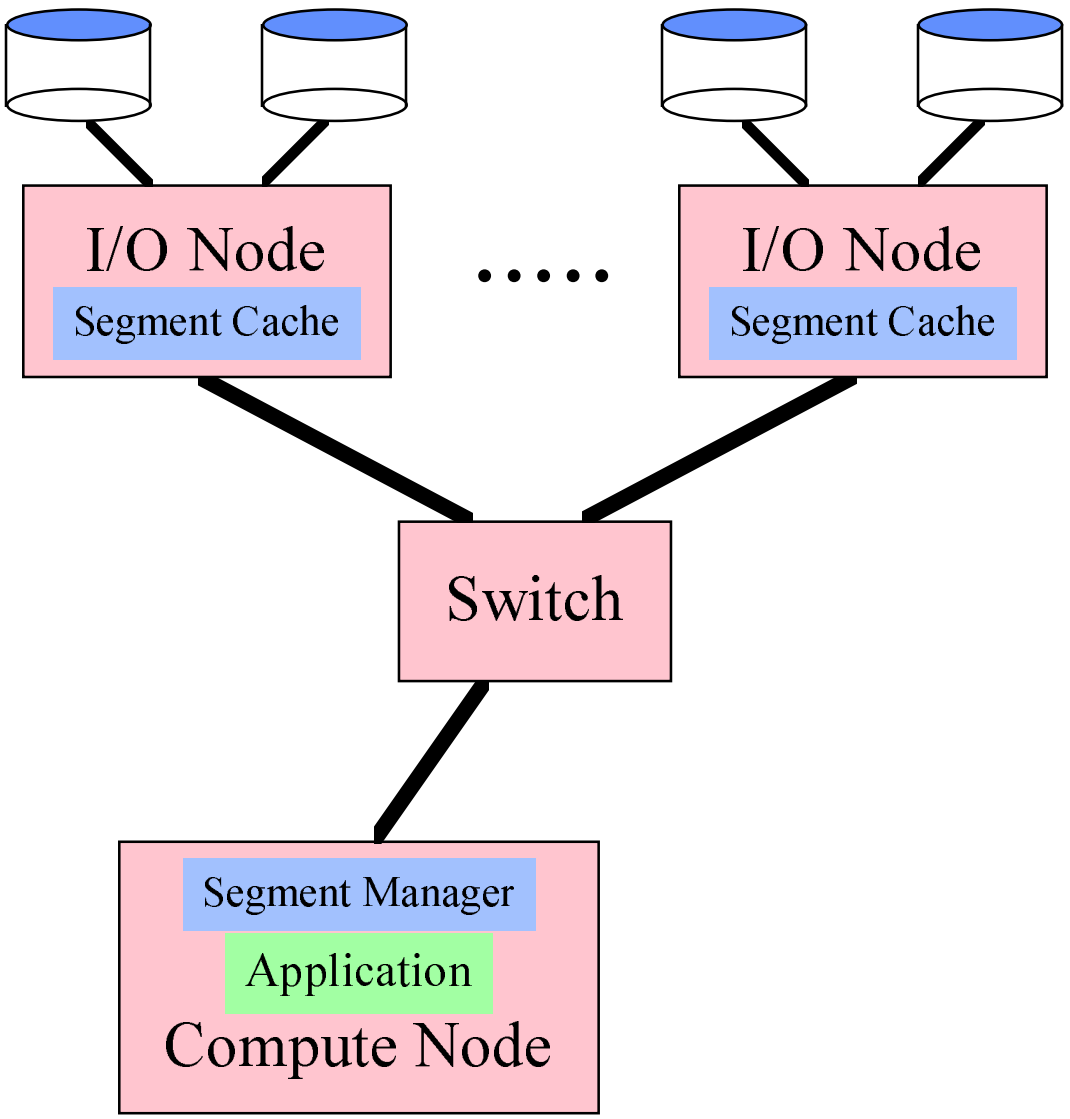


Using pTool



Application sees Segment as mmap'd memory. A page fault occurs if segment is not in memory, The page fault causes the segment to be retrieved from the disk.

Using Fermi-pTool



The application deals with a segment put into real memory. If the correct segment is not present, a segment manager requests that an I/O server read and transmit the needed segment from disk. Writing into the segment is not allowed.

POPM - the next generation

There are several problems with pTool:

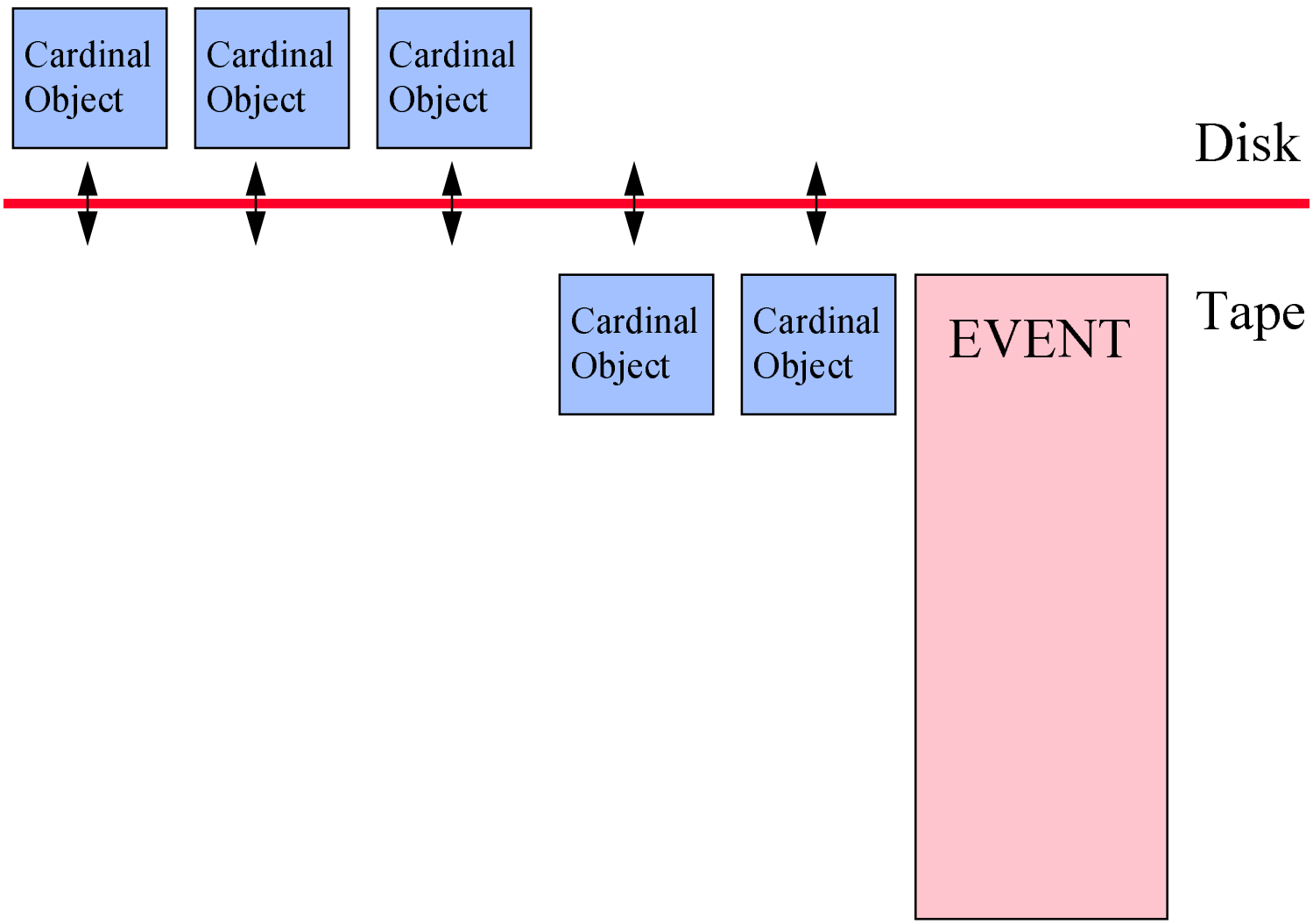
- The original design was not intended for distributed systems
- Many operations are synchronous. Look-ahead and cacheing help somewhat, but make large memory requirements.
- Synchronous aspects particularly impair effective multi-user operation.
- Striping of data is inefficient.
- Several avoidable memory-to-memory copies.

POPM addresses these concerns with:

- Flexible persistent pointers, with fields defined in store metadata.
- Intermediate use of a “physical pointer” to avoid redundant persistant pointer access.
- Distributed implementation

Questions not addressed:

- Non-portable object data
  - Byte ordering problems
  - Record alignment/padding problems
  - Segment size architecture dependence

- “Objects” vs “Structures”
  - Member functions are not persistent; functions which work on one ‘version’ of the data may not work on other versions. May need to always encode and check data version numbers.
- Various other problems using constructors, member functions for persistent objects...

The CAP Model