Parallel File Striping
Design trade-offs

R. D. Hersch, B. Gennart, EPFL

{hersch,gennart}@di.epfl.ch
http://visiblehuman.epfl.ch

Content

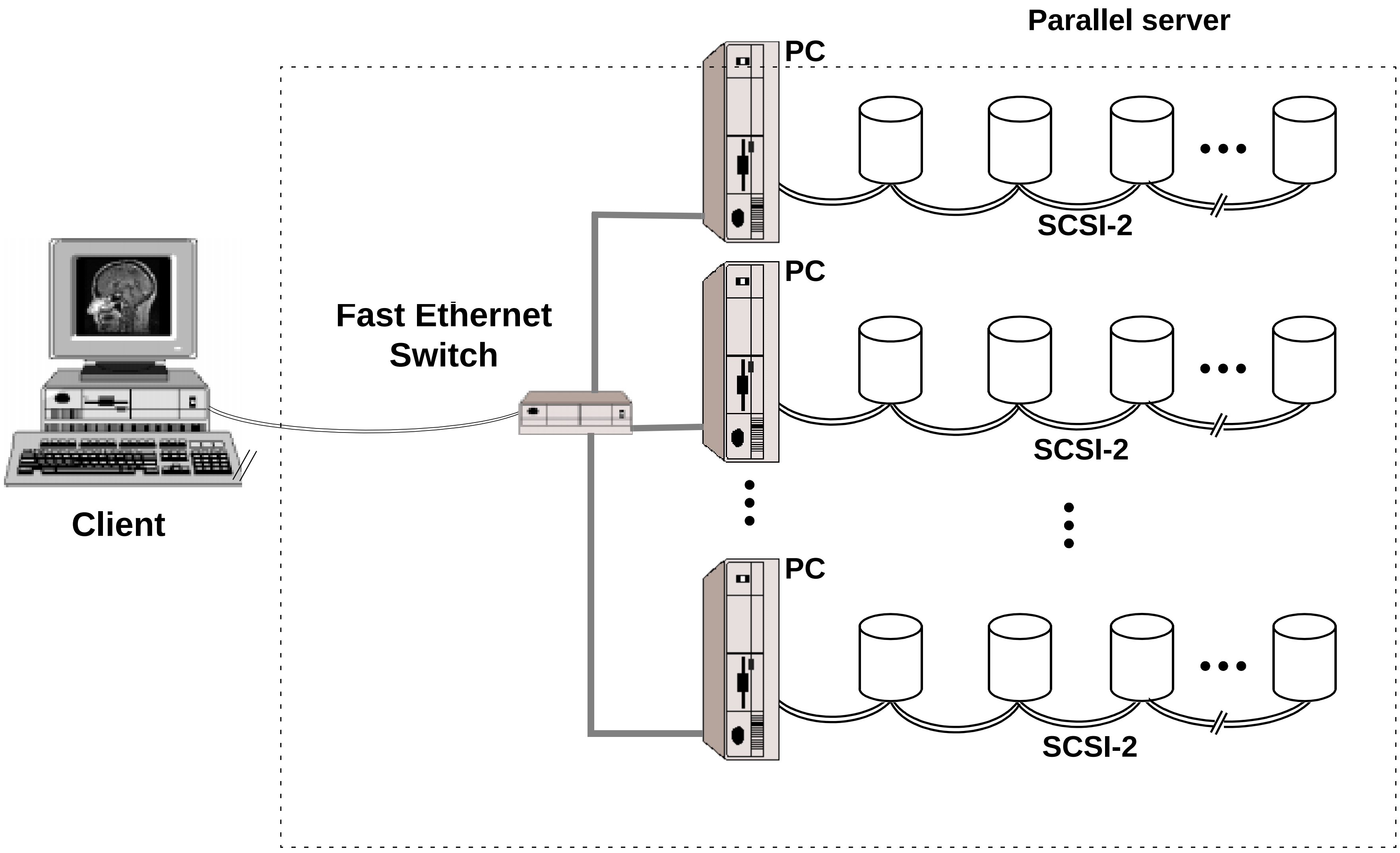
1. Context

- Parallel distributed memory servers
  for I/O - & compute-intensive applications

- Assumptions:
  - Parallel file striping library linked
    to application
  - Local files are standard files (NT, Unix)

- Goal: make maximal use of underlying
  hardware and software :
  => Simultaneous I/O accesses,
     communications and computations
  => Generate asynchronous schedule of
     operations

Considered Architecture



2. I/O Interface at the striped file level

- **Generating asynchronous schedules with CAP (Computer-Aided Parallelization)**

- **Example: The Visible Human Slice Server**
  http://visiblehuman.epfl.ch
  see next slices:
  - **Pipelined parallel slice extraction**
  - **Performances**

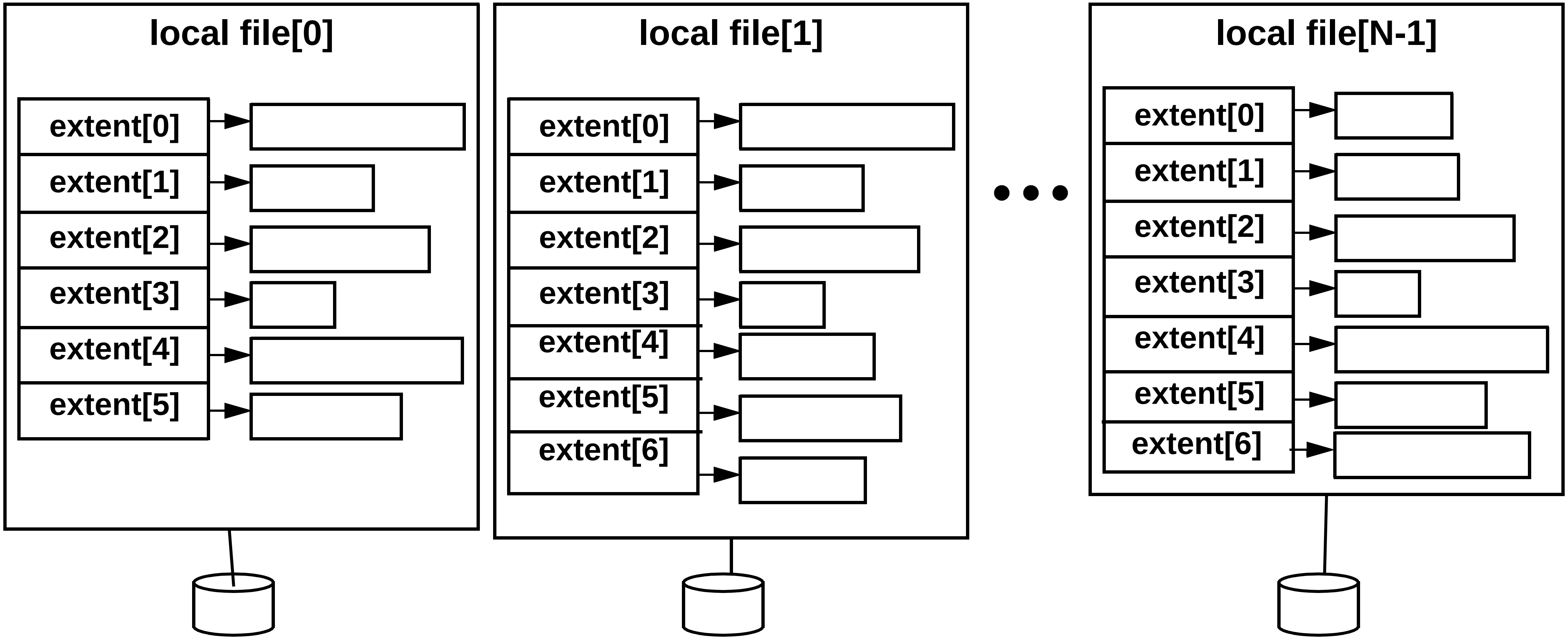
- **Mapping specified by application:**
  Global file position
  -> LocalFileIndex
  -> LocalExtentIndex
  System library specified mapping
  LocalFileIndex -> DiskServerIndex
  DiskServerIndex -> ComputeServerIndex

- **Programmer must deal with 4 indices**
  Advantages:
  - **When possible, computations on same node as disk data**
  - **Simultaneous computations & disk accesses (programmable prefetching strategy)**
  - **Disk directed I/O feasible, i.e. disk server may request data from appl. processes**

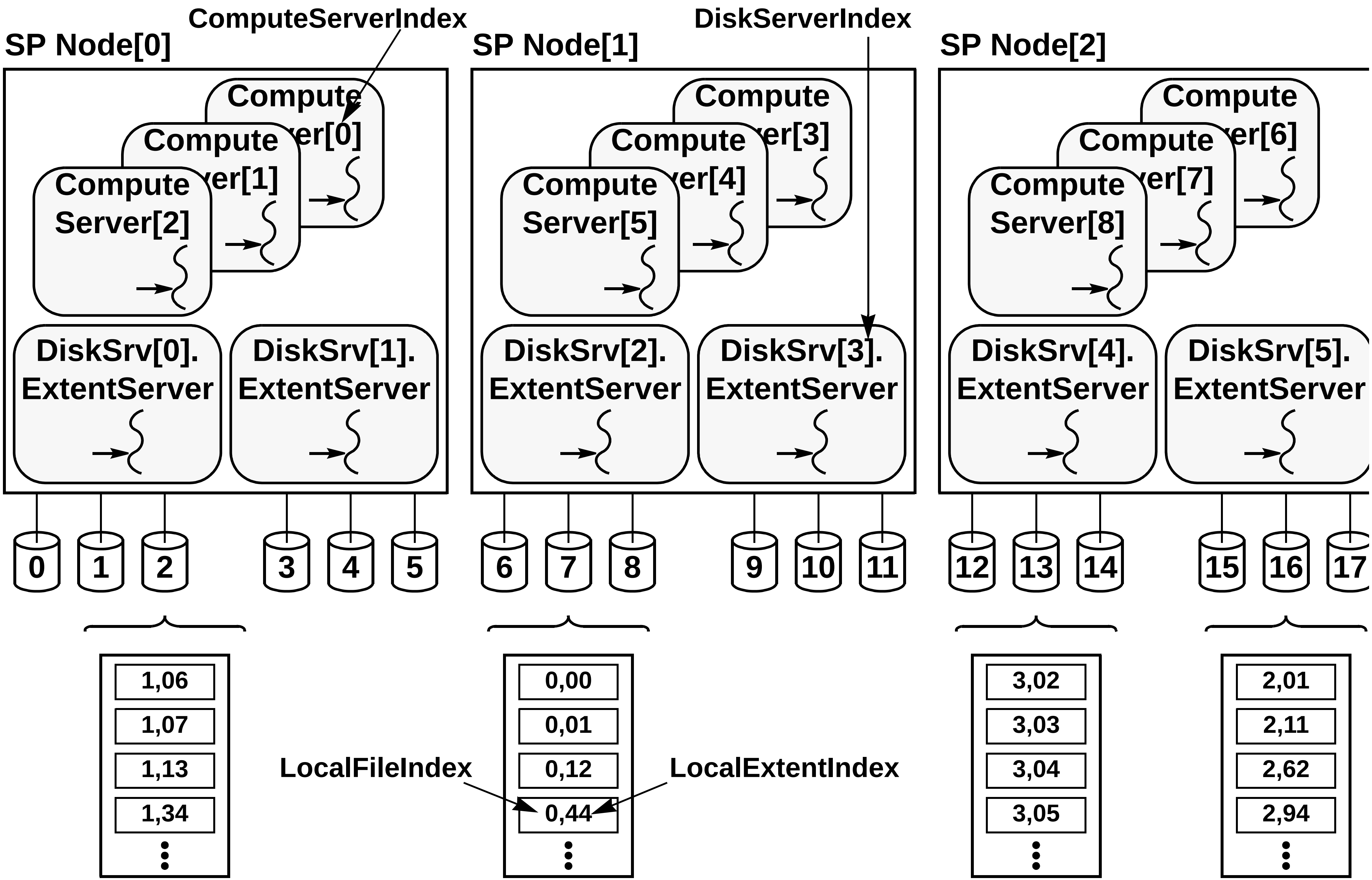
Two-Dimensional Parallel File Structure

extent location given by <LocalFileIndex, LocalExtentIndex>

parallel file declustered across N local files (disks)



LocalFileIndex, LocalExtentIndex, DiskServerIndex & ComputeServerIndex



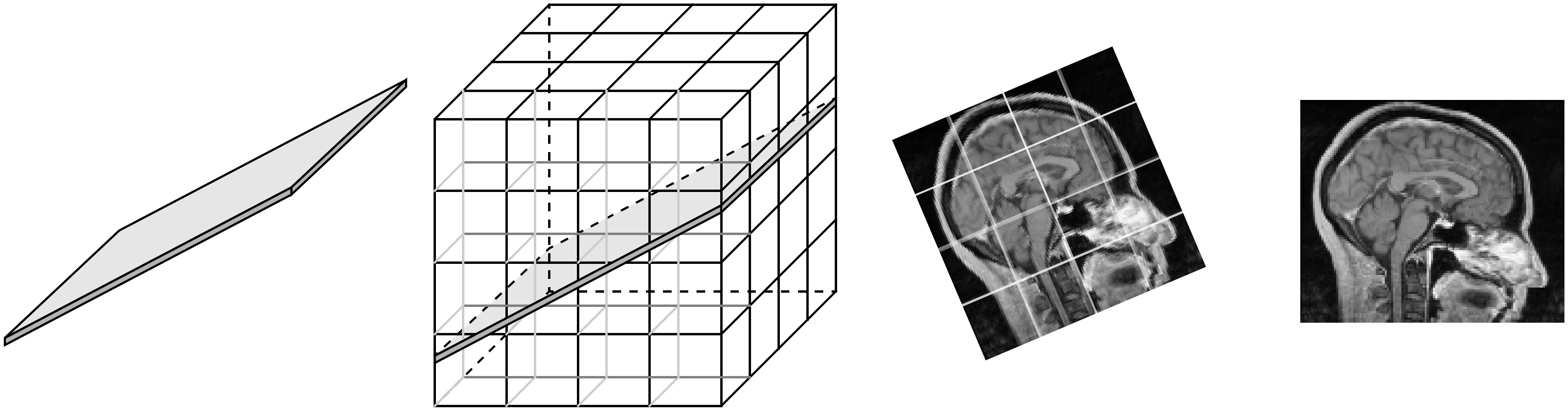
Example: the Visible Human Slice Server

(http://visiblehuman.epfl.ch)

- Human Male: 13 GB of data striped over 60 disks (sub-volumes: file extents)
- Resolution: one anatomical slice/mm
  in each anatomical slice: 3 samples/mm
- Size: 1840 horizontal slices,
  each 2048x1216 pixels

- Software :
  - client sends slice position & orientation
  - server interface asks for subslices
  - server processors read sub-volumes, extract and project slices (resampling)
  - server interface assembles resulting full slice, compresses it and sends it to client

Extraction of Slice Parts from Volumic File Extents



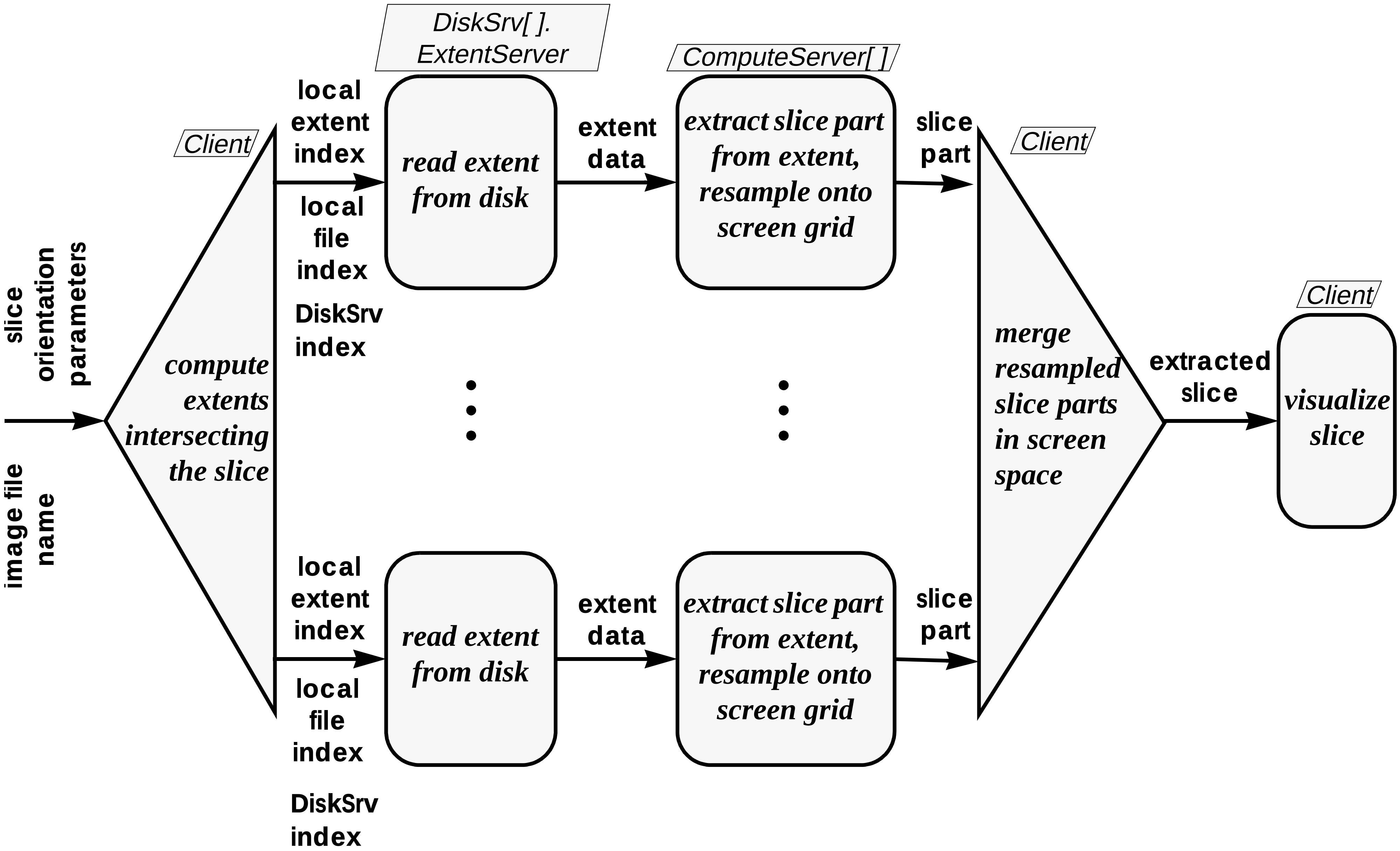
Extraction slice specification

Extraction of the digital slice from the 3D image

Extracted slice parts

Resampled slice parts merged into the final displayable slice

Pipelined Parallel Slice Extraction



CAP Specification of the Pipelined Parallel Slice Extraction

```
leaf operation Ps2ExtentServerT::ReadExtent
  in ExtentReadingRequestT* InputP
  out ExtentT* OutputP;

leaf operation Ps2ComputeServerT::ExtractAndResampleSlicePart
  in ExtentT* InputP
  out SlicePartT* OutputP
{  ...C++ sequential code  }

bool SplitSliceRequest(SliceExtractionRequestT* FromP,
                       ExtentReadingRequestT* PreviousP,
                       ExtentReadingRequestT* &ThisP)
{
  ...C++ sequential code
  return (IsNotLastExtentReadingRequest);
}

void MergeSlicePart(SliceT* IntoP, SlicePartT* ThisP)
{
  ...C++ sequential code
}

operation Ps2ServerT::ExtractSlice
  in SliceExtractionRequestT* InputP
  out SliceT* OutputP
{
  parallel while (SplitSliceRequest, MergeSlicePart, Client, SliceT Output)
  (
    DiskSrv[thisTokenP->DiskServerIndex].ExtentServer.ReadExtent
    >->
    ComputeServer[thisTokenP->ComputeServerIndex].ExtractAndResampleSlicePart
  )
}
```

see http://diwww.epfl.ch/w3lsp/pub/publications/gigaview/captutorial.pdf

Visible Human Slice Extraction Performances

- Max. configuration, caches disabled
  5 server PC's, each with 12 disks

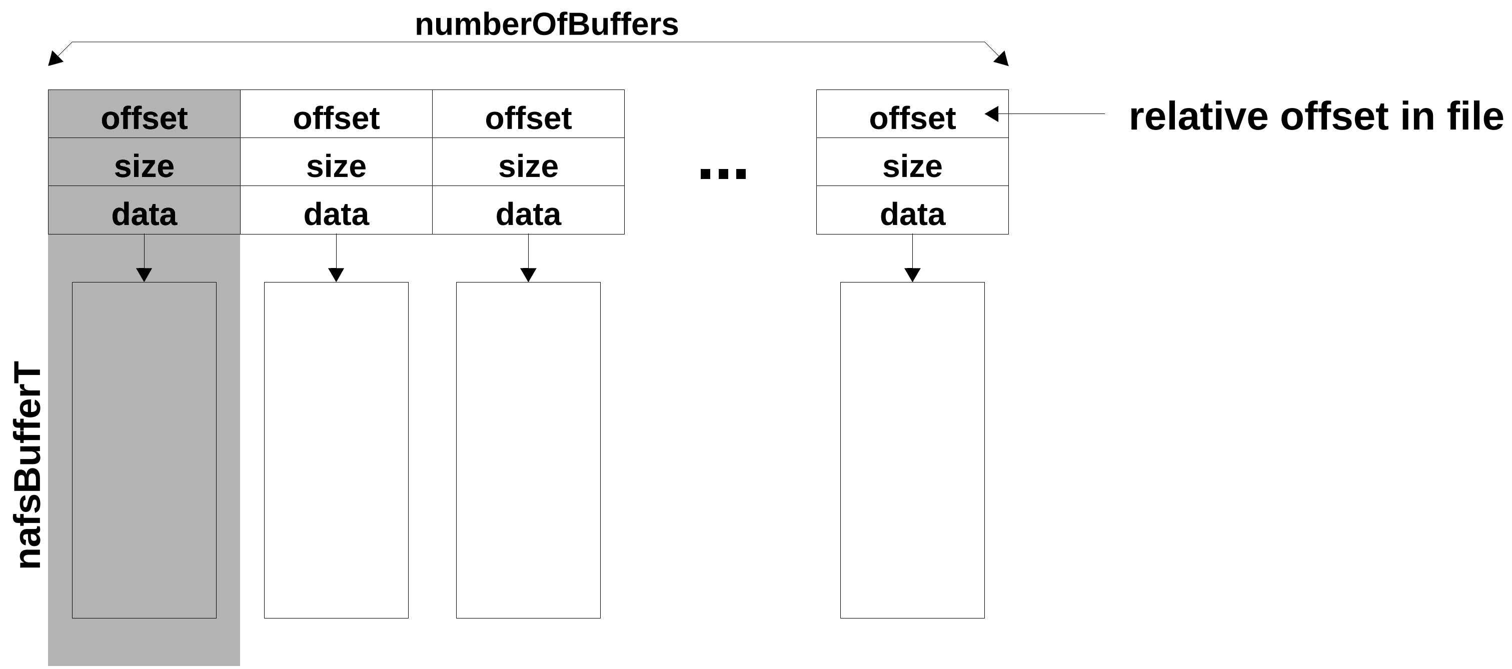
  4.8 extracted slices/s
  client processor utilization: 85%
  => 104 MB/s read from 60 disks
  => mean throughput per disk 1.74 MB/s
  => disks are bottleneck

3. High-level Striped File Interface (NAFS)

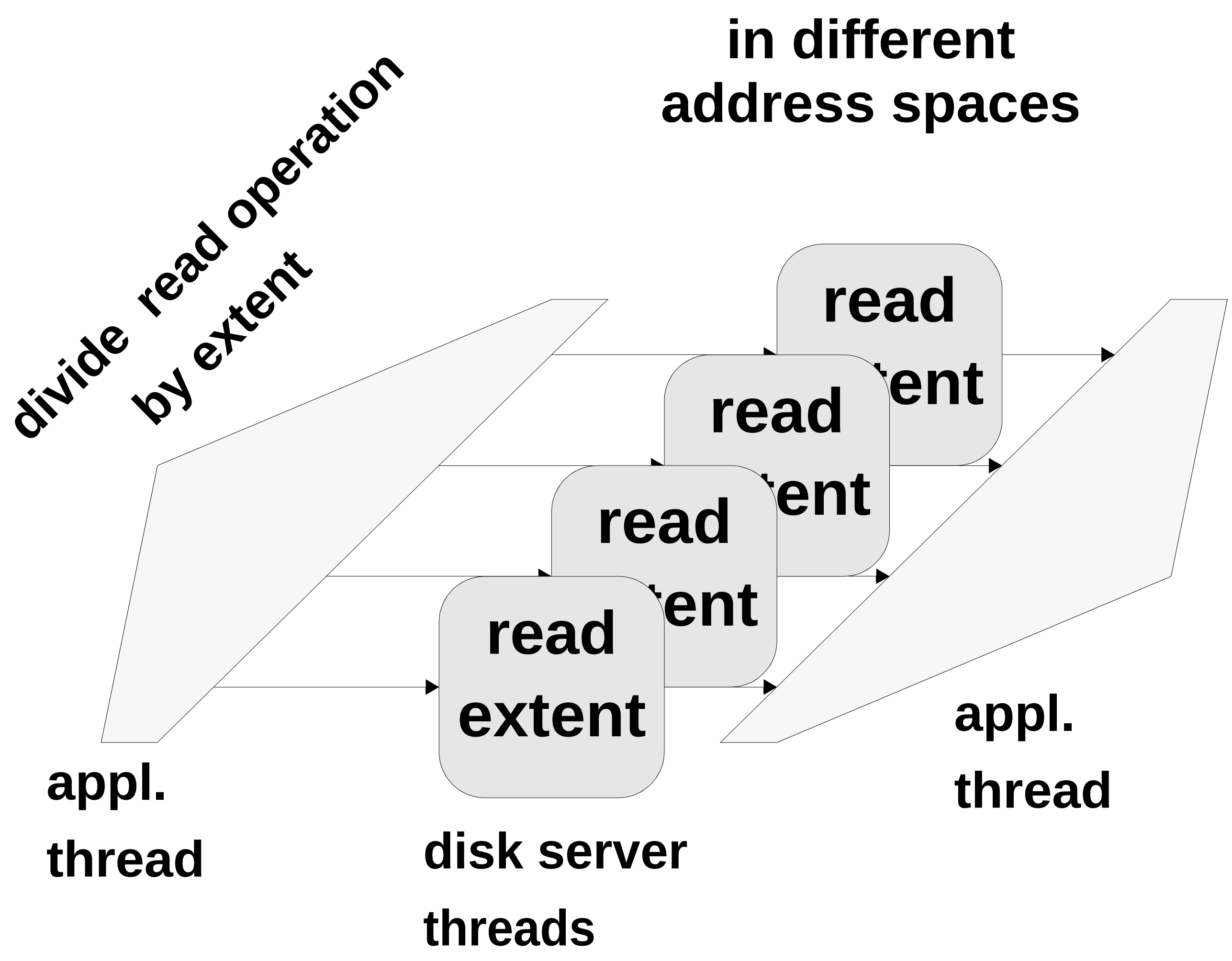
- Interface translates global byte positions to LocalFileIndices, LocalExtentIndices and local byte positions
- User specified stripe unit, striping over a set of user specified paths (disks).
- Read, Write, CollectiveRead, CollectiveWrite of a set of contiguous data buffers
- Assessment: Simple interface, but no direct knowledge about locality of data

High-level striped file interface (NAFS)

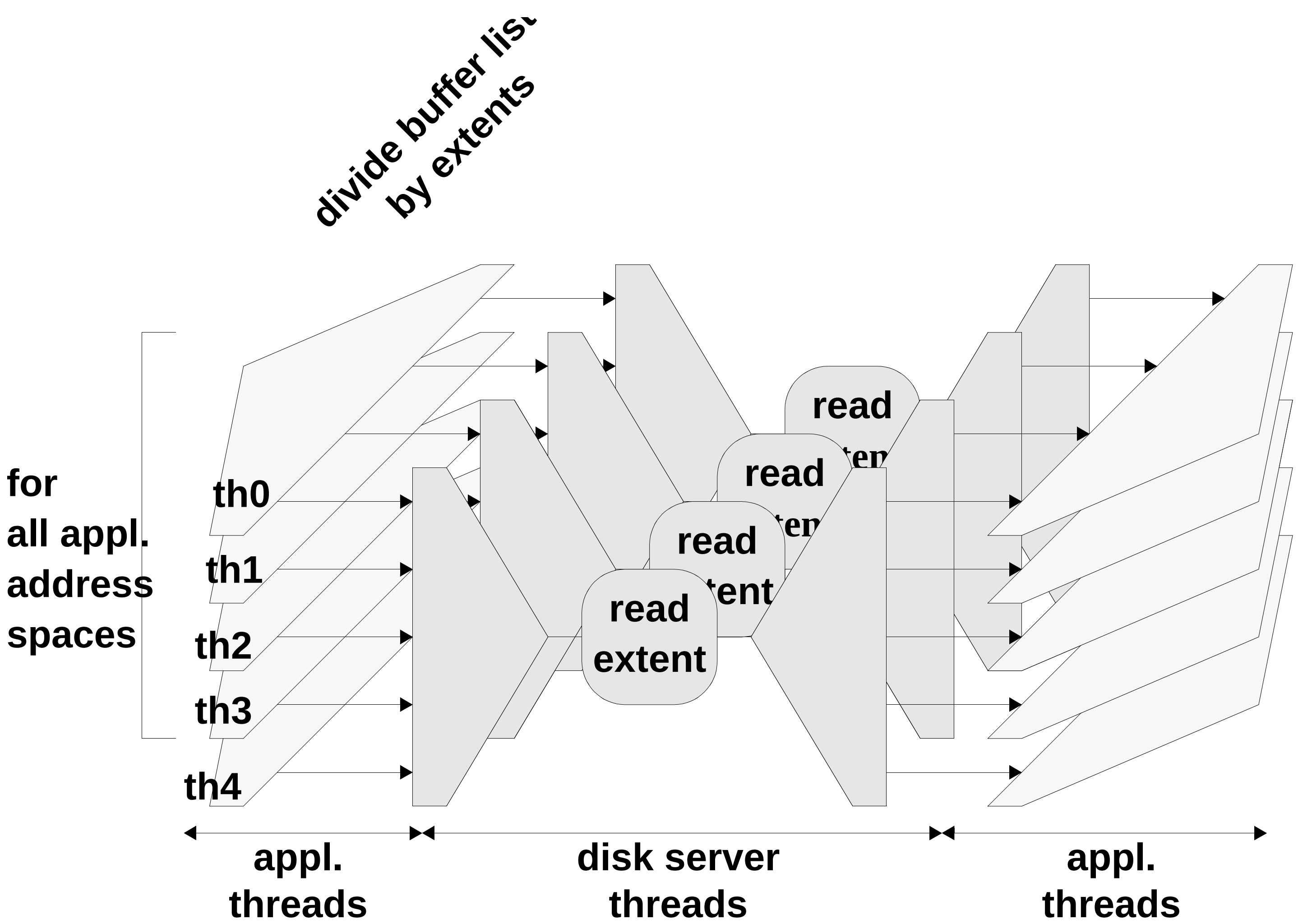
```
nafsFileT* nafsOpen (char* fileNameS, char openMode) ;
int nafsClose (nafsFileT* fileP) ;
int nafsCreate (char* fileNameS) ;
int nafsDelete (char* fileNameS) ;
int nafsWrite (nafsFileT* fileP, UINT64_T absoluteOffsetInFile,
              UINT32_T numberOfBuffers, nafsBufferT* buffersP) ;
int nafsRead  (nafsFileT* fileP, UINT64_T absoluteOffsetInFile,
              UINT32_T numberOfBuffers, nafsBufferT* buffersP) ;
int nafsCollectiveWrite (nafsFileT* fileP, UINT64_T absoluteOffsetInFile,
                      UINT32_T numberOfBuffers, nafsBufferT* buffersP) ;
int nafsCollectiveRead (nafsFileT* fileP, UINT64_T absoluteOffsetInFile,
                      UINT32_T numberOfBuffers, nafsBufferT* buffersP) ;
```



Read operation



Collective read operation



4. Performance of the high-level Striped File Interface

- **Configuration:**
  4 Bi-PentiumPro PC's, with 8 disks (5400 rpm), interconnected by Fast Ethernet switch

- **Writes to local disks: 8 x 2.5MB/s => 20MB/s (32KB data chunks), no much processing power used**

- **Collective write: each processor writes to each disk many 2KB data chunks**

  max sustainable Fast Ethernet throughput per PC: 4MB/s (100% processor utilization)

  => 2MB/s for sending out data from application
  2 MB/s for receiving data to the disks

  => 2MB/s per node global write throughput
  half the processing power used

5. Conclusions

- As long as TCP-IP is used, minimize accesses to remote disks (low-level striped file interface)

- Even for I/O
  fast communication network is required
  => Socket based interface, with flow control but without TCP-IP processing overhead

- High-Performance File Server:
  TCP-IP data transfers considerably reduce effective RAID access bandwidth
  => during computations, parallel I/O across a large number of nodes is preferable (SwissTx: 3MB/s per node expected)

- After computation session,
  save striped files to file server
  pmv, pcp: parallel move or parallel copy

- In case of processor/disk failure, current results are lost; relaunch computation

- Option: delayed generation of a parity file