Building A High Performance Parallel File System Using Grid Datafarm and ROOT I/O

Y.Morita, H.Sato, Y.Watase, A.Manabe (KEK)

O.Tatebe, S.Sekiguchi (AIST)

S.Matsuoka (Titech)

T.Kobayashi (ICEPP)

N.Soda (SRA)

A.Dell'Acqua (CERN)

Powered by SuperSINET

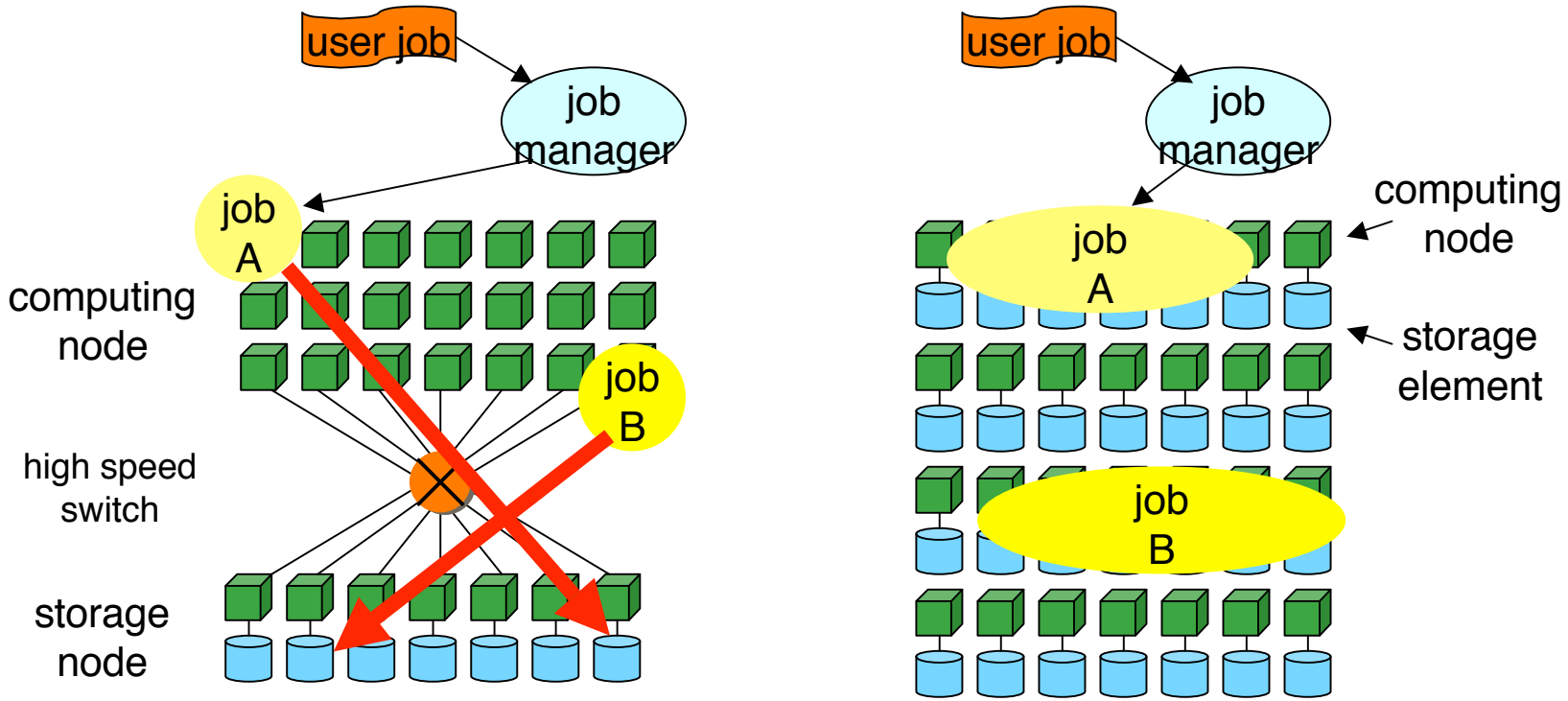
The “Challenge”

- Petabyte-scale data analysis in world-wide collaboration
- Thousands, or tens of thousands CPUs and storage elements as a “system”
- Network Bandwidth outperforming the Moore’s law:
  “LFN” -- a few 100msec RTT with Gigabit network
  ⇒ needs multiple stream
- Management of the job workflow of thousands of users

The “Solution”

- Exploit the HENP event data access locality
- Distributed I/O + Distributed Analysis
  - File Affinity Scheduling
- “Cluster of Cluster” file system
  - File replica with striped file transfer
- Security: Grid authentication

Data Access Locality

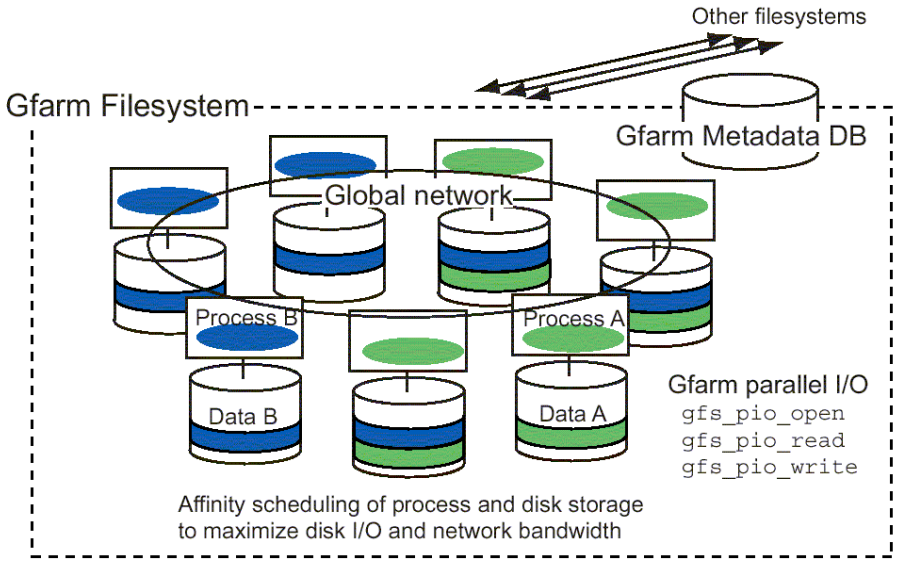


Exploit the data access locality as much as possible

Software Suite

Gfarm

- Grid Data Farm : 1st prototype presented at GGF1, March 2001
- Collaboration to develop Grid middleware: AIST, Titech, KEK, ICEPP
- Parallel I/O: exploit the data access locality, store and access files by "fragments"
- Parallel Job: program runs on the nodes where the file fragments reside: "owner computes"
- Programs and file fragments, job history, data checksum, file replication are managed with MetaDB
- Fragments are replicated for backup and load balancing
- User sees the fragmented files as a single Gfarm URL
- Provide system call hooks for open(), close(), read(), write() etc
- Authentication: Globus GSI and/or Shared Private Key



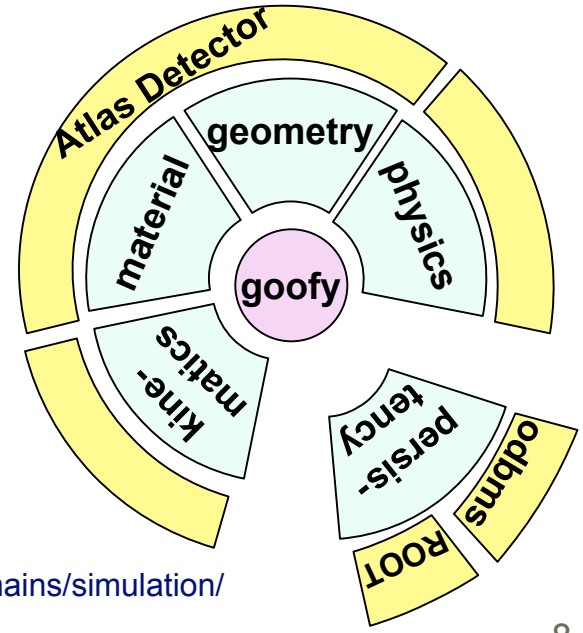
http://datafarm.apgrid.org/

Gfarm: how it looks to user

```
% gfreg  your_prog  gfarm:my_prog
% gfls -l
-rwxr-xr-x morita   *            531590 Mar 24 21:06 my_prog
% gfrun -N 3 gfarm:my_prog -f gfarm:my_prog_out
% gfls -l
-rwxr-xr-x morita   *            531590 Mar 24 21:06 my_prog
-rw-r--r-- morita   *     135291469824 Mar 25 06:52 my_prog_out
% gfwhere gfarm:my_prog_out
0: pad001
1: pad002
2: pad003
```

FADS/Goofy: a light-weight framework

- Framework for Autonomous Detector Simulation/ Geant4-based Object-Oriented Folly
- Thin and versatile framework for Geant4 simulation
- Can load new service plug-ins at runtime
- Utilize Geant4 services as much as possible
  - Visualization, User Interfaces, ...
- Supports HBOOK and ROOT for histogramming
- Enables rapid prototyping of the detector code
- Separation of abstract I/F part and technology dependent part
- Persistency: Objectivity/DB and ROOT I/O

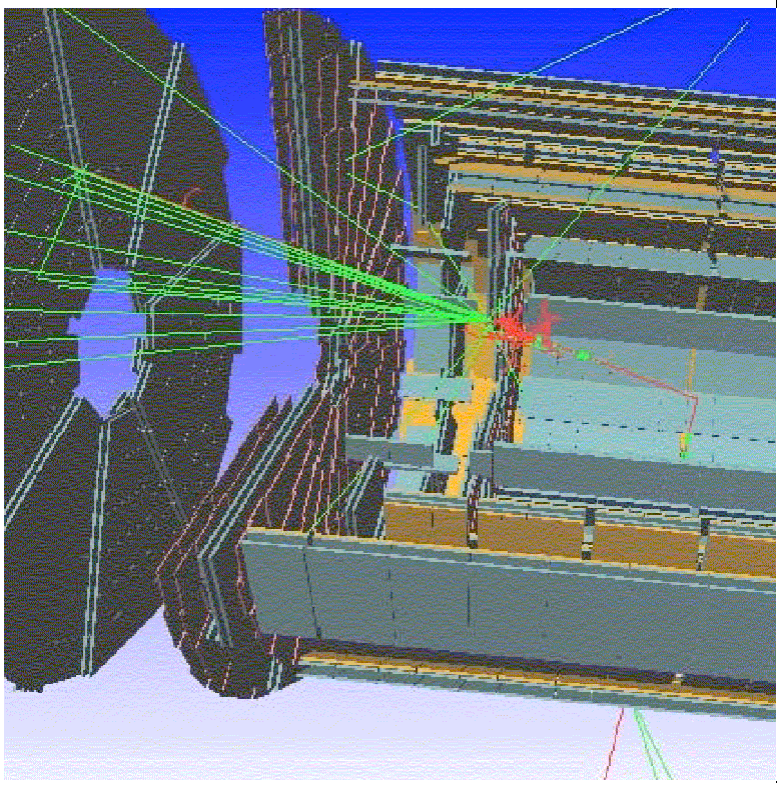


http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/domains/simulation/

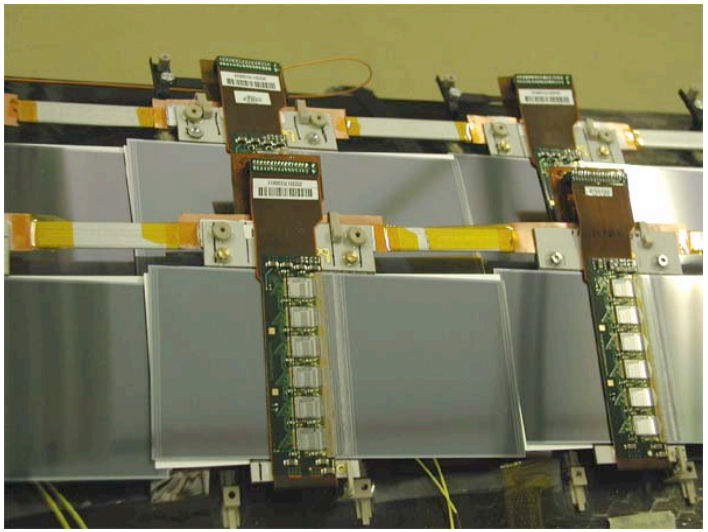
FADS/Goofy in Atlas

- Used in sub-detector software developments and physics validations
- Detector modules also run in Atlas mainstream framework (Athena)

- Testbed for ROOT I/O
- Bandwidth Challenge in SC2002
- Generated 10^6 fully simulated higgs 4 events in 2 days with ~ 400 CPU

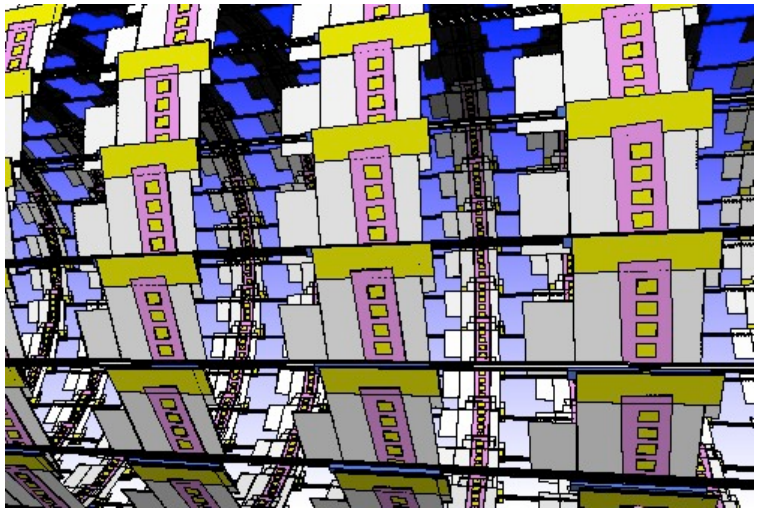


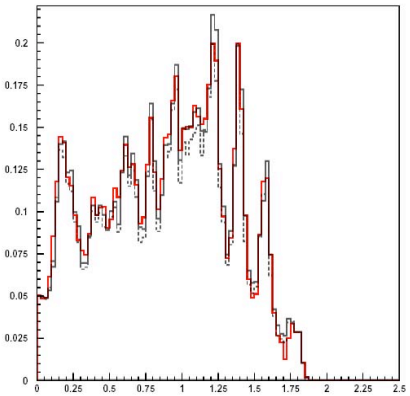
Radiation Length Study w/ FADS/Goofy



Atlas Silicon Tracker (SCT)

Geant4 SCT volumes by
A.Dell'Acqua, Y.Tomeda et al





Consistency check between G3 and G4

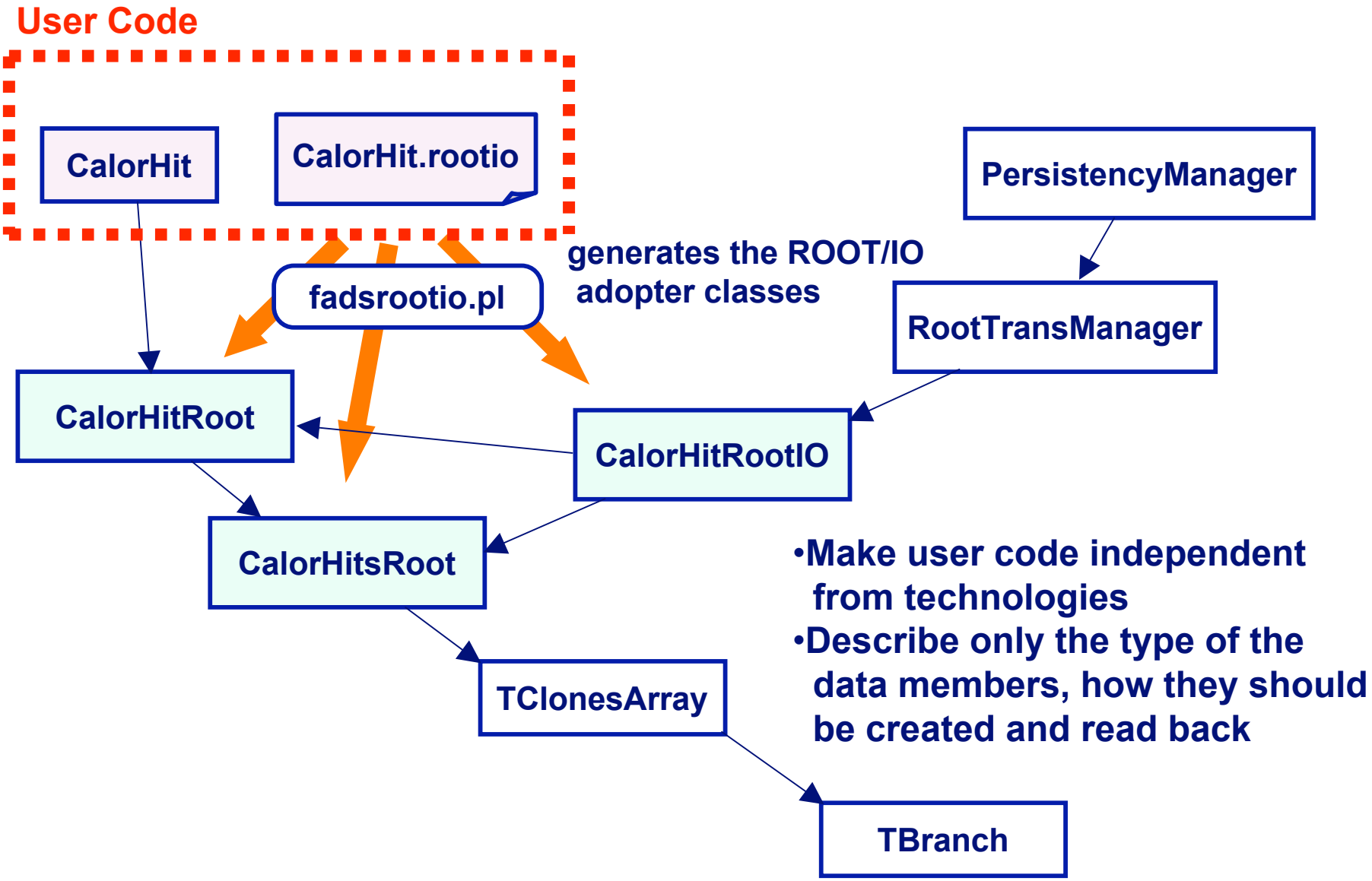
Light-weight, portable, complete framework helps for education

ROOT I/O in FADS/Goofy

Persistency in FADS/Goofy

- Users of FADS/Goofy design transient detector response (collections of hit/digit class)
- Users also provide a simple data description file (*.rootio) for persistent data store
- A perl script *fadsrootio.pl* generates FADS ROOT I/O adopter classes (*HitRoot.hh, *HitsRoot.hh, HitRootIO.hh and *.cc)
- Hits/Digits collections are stored/retrieved into the ROOT branches with TClonesArray

Also provided as Geant4 persistency example as g4rootio.pl *in Geant4 5.0*


User Code
CalorHit
CalorHit.rootio
fadsrootio.pl
generates the ROOT/IO
adopter classes
PersistencyManager
RootTransManager
CalorHitRoot
CalorHitRootIO
CalorHitsRoot
TClonesArray
TBranch
•Make user code independent from technologies
•Describe only the type of the data members, how they should be created and read back

Example CalorimeterHit.hh

```
class Pers01CalorHit : public G4VHit
{
   public:
     Pers01CalorHit();
    ~Pers01CalorHit();
      .....
   public:
     void AddAbs(G4double de, G4double dl) {EdepAbs += de; TrackLengthAbs += dl;};
     void AddGap(G4double de, G4double dl) {EdepGap += de; TrackLengthGap += dl;};

     G4double GetEdepAbs()    { return EdepAbs; };
     G4double GetTrakAbs()    { return TrackLengthAbs; };
     G4double GetEdepGap()    { return EdepGap; };
     G4double GetTrakGap()    { return TrackLengthGap; };

 private:
     G4double EdepAbs, TrackLengthAbs;
     G4double EdepGap, TrackLengthGap;
};
```

Example CalorimeterHit.rootio

```
set class_name Pers01CalorHit
set collection_class Pers01CalorHitsCollection
set collection_base_class G4VHitsCollection
set sdet_name Pers01CalorHit
set array_io_base VPHitsCollectionIO
set catalog HCIOentryT
set global_declaration
  class @class_name@;    // forward declaration
..
set add_header_src
  @class_name@.hh
  G4ThreeVector.hh
  G4RotationMatrix.hh
..
set member
  @float@  EdepAbs;
  @float@  EdepGap;
  @float@  TrackLengthAbs;
  @float@  TrackLengthGap;
..
```

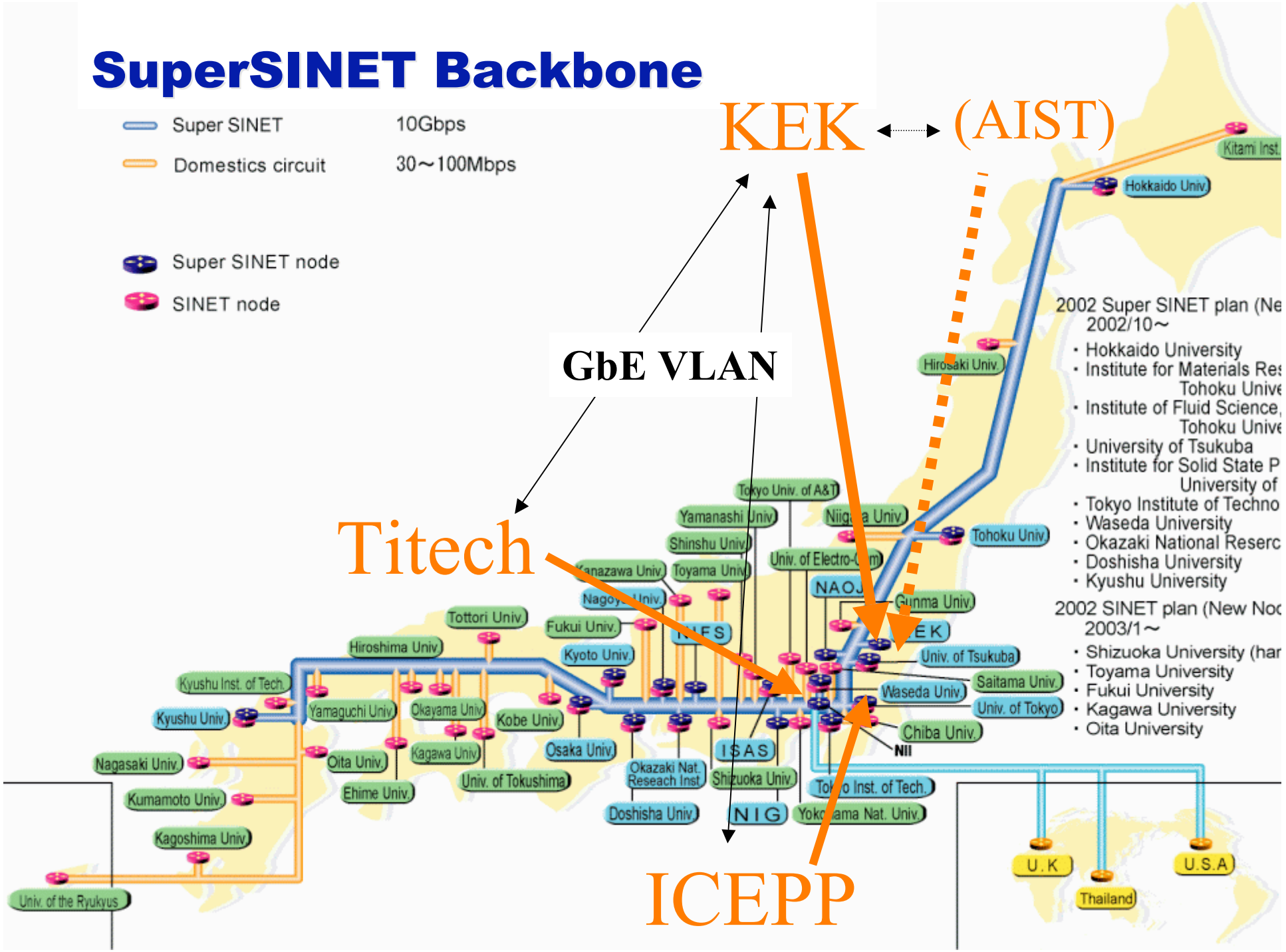
```
set constructor
  @class_root@(@class_name@* hit)
  {
    // copy data members of transient hit
    EdepAbs = hit->GetEdepAbs();
    EdepGap = hit->GetEdepGap();
    TrackLengthAbs = hit->GetTrakAbs();
    TrackLengthGap = hit->GetTrakGap();
  }
..
set method
  @class_name@* @make_transient@()
  {
    // create a transient class
    @class_name@* hit = new @class_name@();

    hit->AddAbs(EdepAbs, TrackLengthAbs);
    hit->AddGap(EdepGap, TrackLengthGap);

    return hit;
  }
```

ROOT I/O Testbed

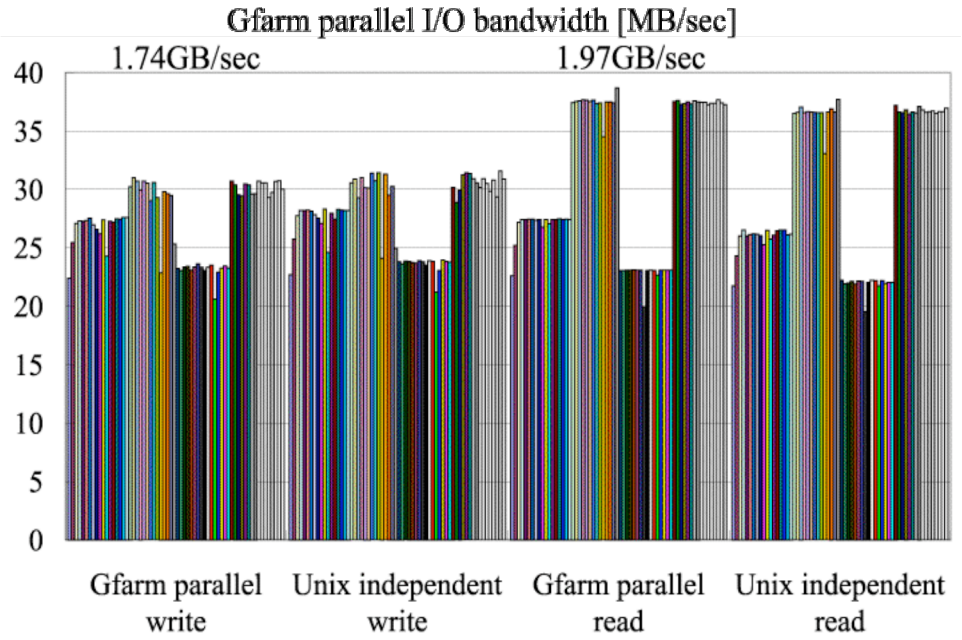
SuperSINET Backbone



Presto-III PC Cluster @ Titech

- Collaboration with AMD, Bestsystems Co., Tyan, Appro, Myricom
- Dual 256 node/512 proc AthlonMP 1900+ (1.6Ghz) Rpeak 1.6 TeraFlops
- AMD 760MP Chipset
- Full Myrinet 2K network
- 100TB Storage for storage intensive/ DataGrid apps
- June 2002
  47th Top 500, 716GFlops
  2nd Fastest PC cluster at the time

Gfarm raw I/O benchmark

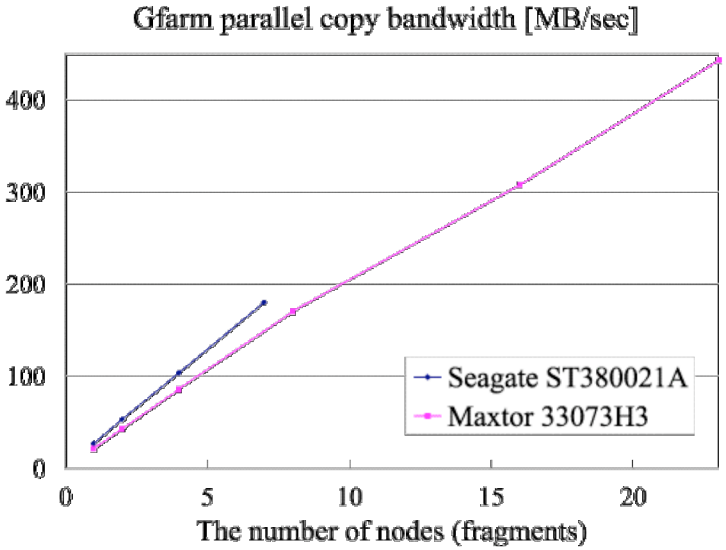



```
write_test(char *fn, void *buf, int size)
{
   GFS_File gf;
   gfs_pio_create(fn, GFS_FILE_WRONLY, mode, &gf;);
   gfs_pio_set_view_local(gf, lflag);
   gfs_pio_write(gf, buf, size, &np);
   gfs_pio_close(gf);
}
```

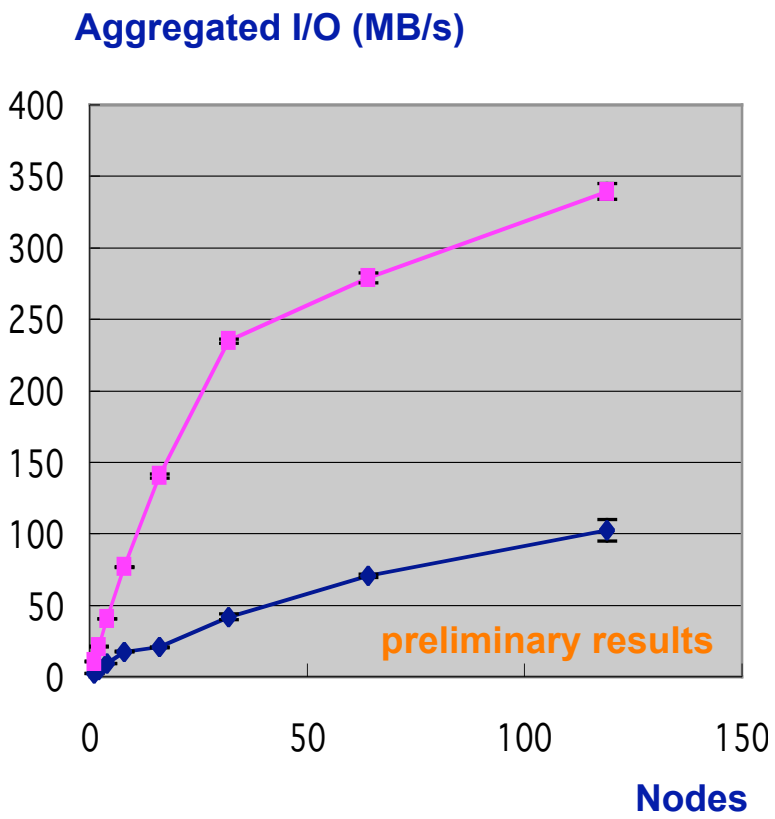
64 nodes, 640 GB file

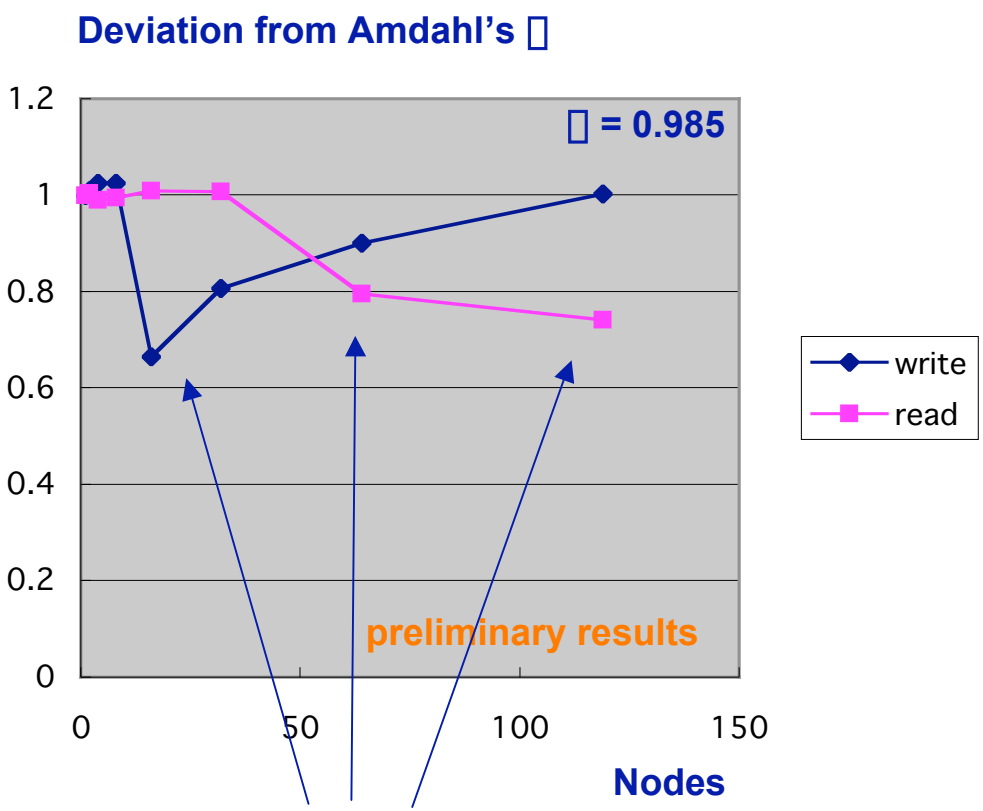
File Replication of 10 GB file fragments through Myrinet 2000

443MB/s at 23 parallel streams



ROOT I/O speed up curve



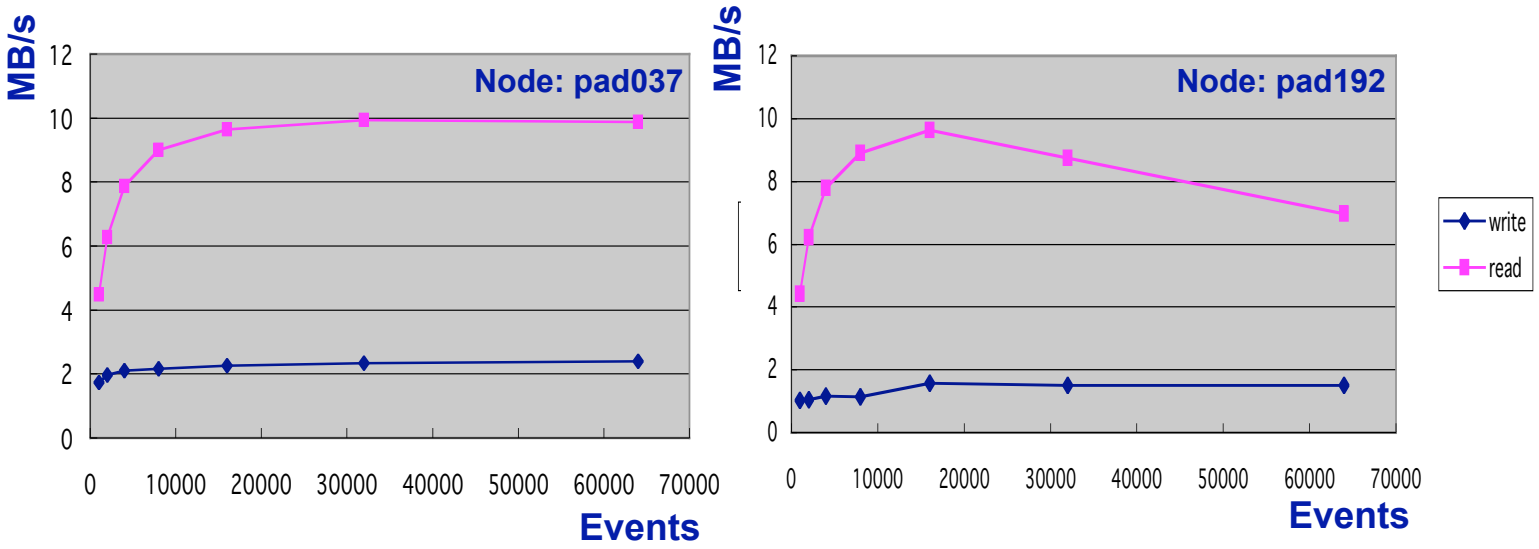


(~ 57MB/fragment, 3.5M hit classes) x (# of nodes)

Speed up is dragged by several "slow" nodes

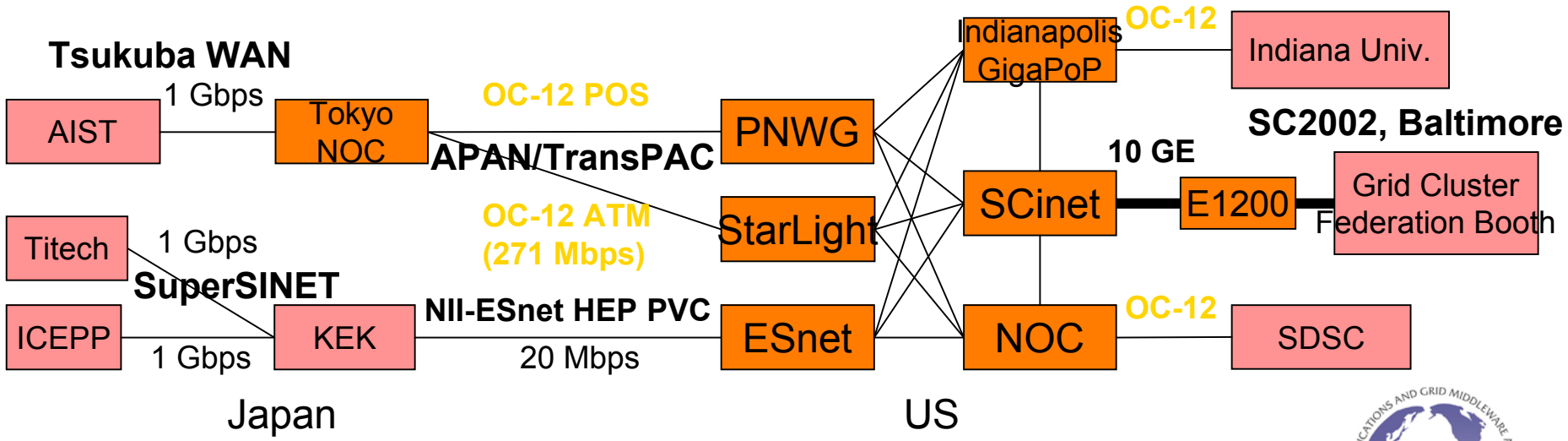
Node behaviors

- Each node gives different performance behaviors
  - free memory, disk fragmentation, cylinder# of the file...
- Increasing the number of nodes is a good “screening” for the node performance test
- Limits of the Gfarm architectural bottlenecks still not reached nor measured... work in progress



Bandwidth Challenge at SC2002

Testbed for SC2002



Total bandwidth from/to SC2002 booth: **2.137 Gbps**

KEK Titech AIST ICEPP SDSC Indiana U SC2002

Total disk capacity: **18 TB**, disk I/O bandwidth: **6 GB/s**

Peak CPU performance: **962 GFlops**

Ganglia Cluster Toolkit

Cluster Report for Wed, 13 Nov 2002 02:41:29 +0900

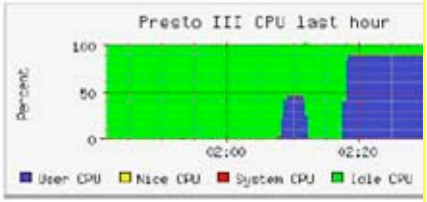
Get Fresh Data

Metric load_one Last hour Sorted descending Physical View

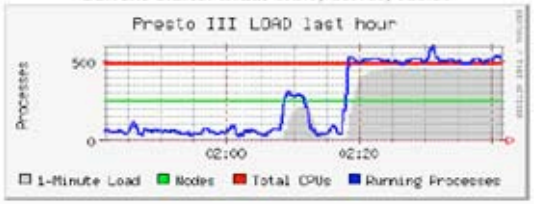
Grid > Presto III > --Choose a Node

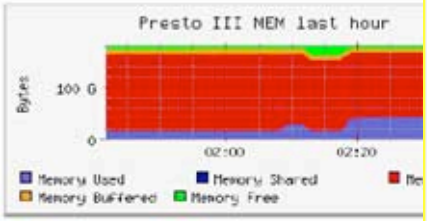
Overview of Presto III

There are 245 nodes (490 CPUs) up and running.
There are 12 nodes down.

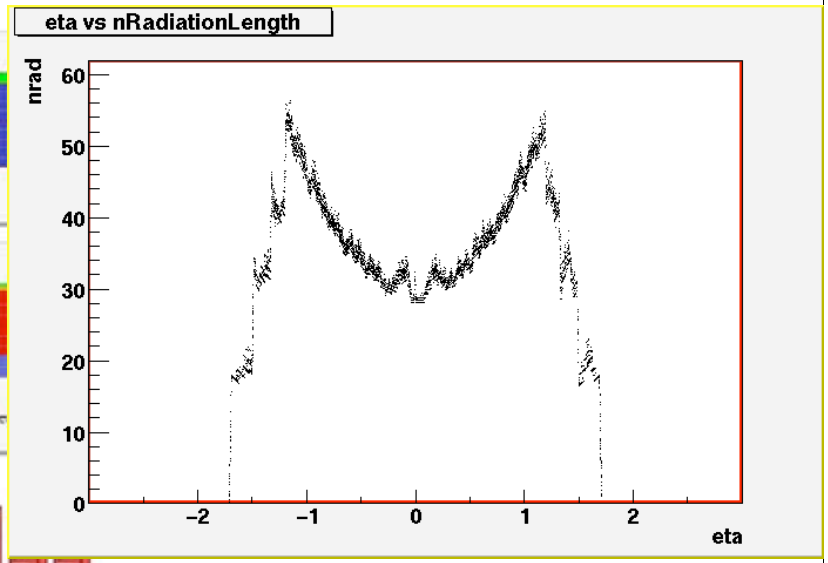


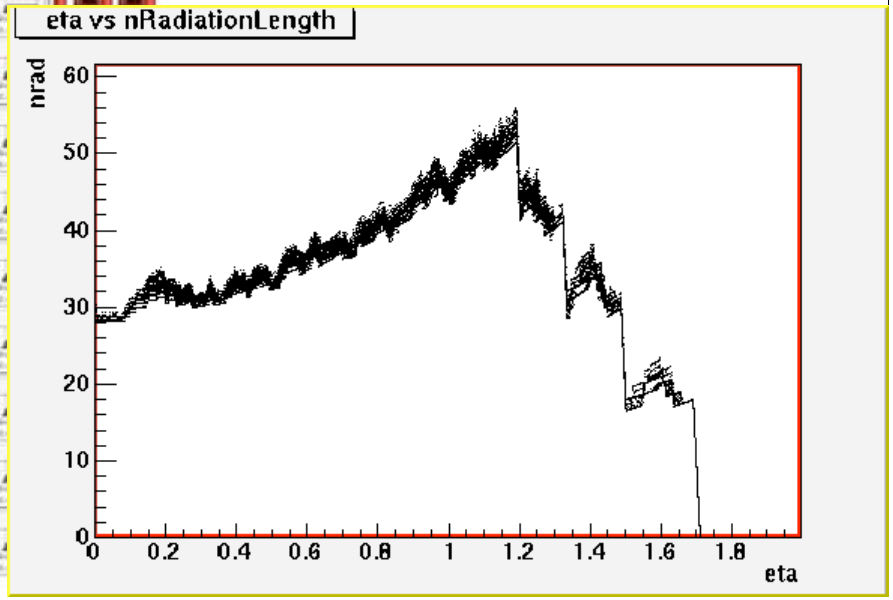
Current Cluster Load: 451.9, 440.46, 318.29



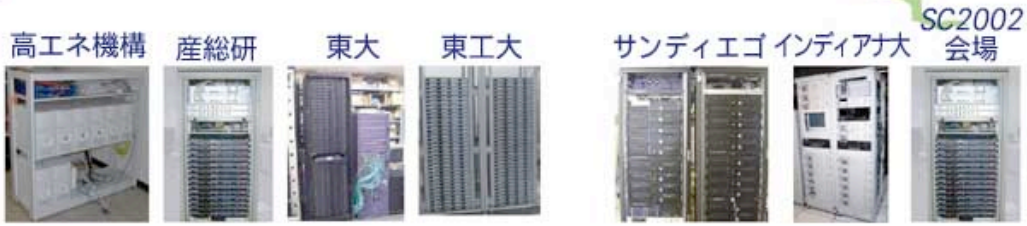


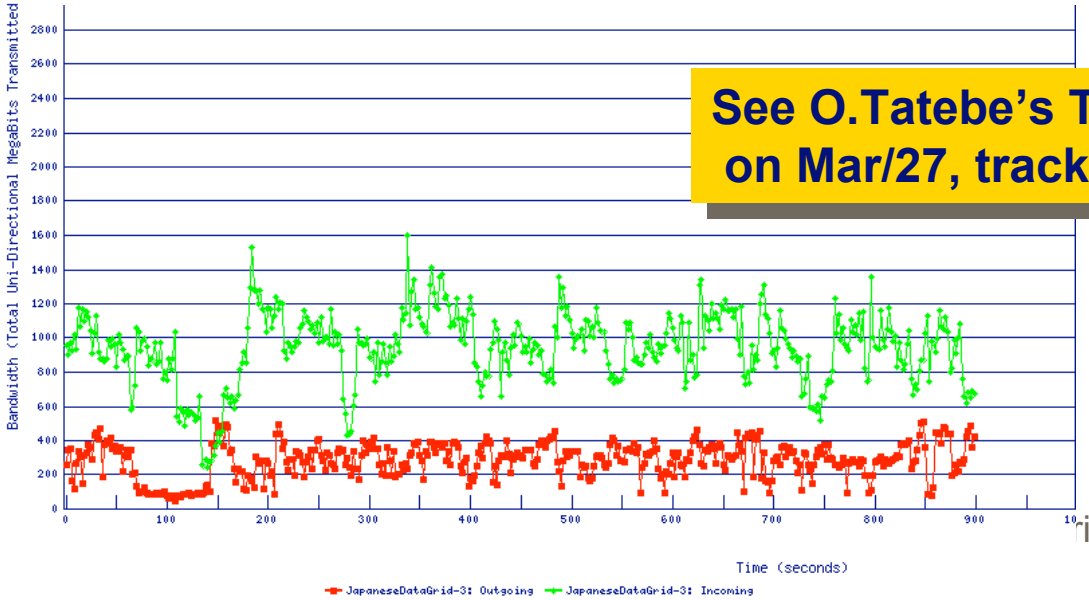
Snapshot of Presto III | Legend

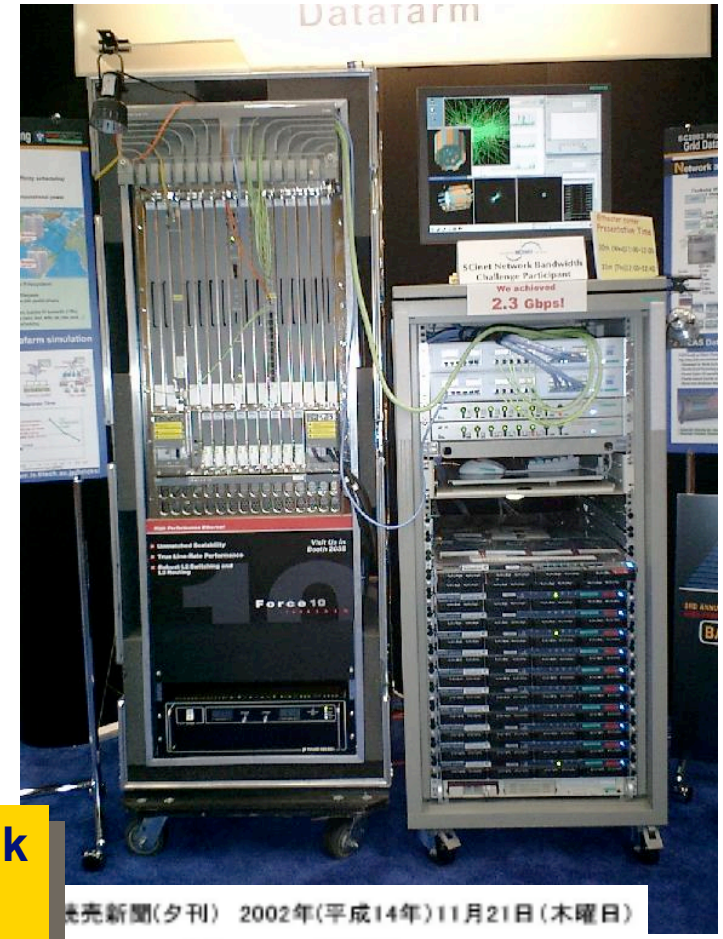




Cluster and Network setting for SC2002 Bandwidth Challenge







See O.Tatebe's Talk on Mar/27, track 1

読売新聞(夕刊) 2002年(平成14年)11月21日(木曜日)

◆毎秒8800万文字分のデータ転送

産業技術総合研究所は21日未明、複数のパソコンを接続して一台のコンピューターとして機能させ（PCクラスタ）、さらにこのPCクラスタを複数統合する新技術を開発、日米間の超大規模データ転送実験に成功したと発表した。

同研究所は新開発のソフトウエアで、同研究所や高エネルギー加速器研究機構、東京大学、米インディアナ大学など日米7拠点、計190台のパソコンを統合したPCクラスタを編成、日米間で1秒当たり7.07ギガ・ビットのデータ転送速度を達成した。これは毎秒8800万文字、CDなら一枚を5・7秒で転送する速さになる。

Some Lessons

- Distributing and installing the testbed package to different management regime is a major “challenge”
  - Many explicit and “implicit” prerequisites: Different OS/Linux flavors, gcc versions, available memory/storage resources, different security/accounting policies, baseline package selections (eg. X11, OpenGL etc), “sudo” access for minor “day to day” configurations, etc...
  - Different testbed package often leads to different prerequisites
  - automated test runs and face-to-face discussions helps
- Bug tracking and a feedback mechanism for the code fixes and redistributing is a “must”
  - spontaneous access to the central code repository from here and there
  - find a bug in one system □ fix it □ test it □ test it everywhere □ fix it □ test it □ test it everywhere □ ...
- Stable/dependable high speed network is our “life-line”
  - needs “hot-line” to the all NOC managers/operators 24/7
- Start planning and organizing your testbed as early as possible

Conclusions

- “File affinity scheduling” provides parallel processing capability for both I/O-bound and CPU-intensive jobs
- ROOT I/O tools (fadsrootio.pl and g4rootio.pl) have been developed for FADS/Goofy persistency
- ROOT I/O module works with Gfarm parallel file system using the system call hook, without a changes to the ROOT package
- Calorimeter Hits ROOT I/O has achived 102 MB/s write and 340 MB/s read using 119 nodes
- Measurements of the speed up curve is limited by the different behavior of the nodes-- work in progress
- ROOT I/O files has been successfully replicated at 2.286 Gbps using the SC2002 Bandwidth Challenge testbed with 12 nodes ~ 190Mbps/node (see O.Tatebe’s Talk)

Special Thanks to ...

- NII SuperSINET NOC
- IMnet NOC
- APAN TransPAC NOC
- SDSC, esp. P.Papadopoulos
- Indiana U., esp. R. McMullen, J. Hicks, C. Robb
- KEK Network Group
- Titech Matsuoka-lab
- Force 10 Networks, Inc.