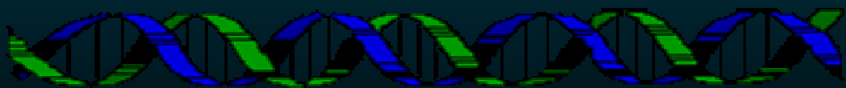
A Case Study of Parallel I/O for Biological Sequence Search on Linux Clusters

Yifeng Zhu, Hong Jiang, Xiao Qin, David Swanson

Department of Computer Science and Engineering

University of Nebraska – Lincoln

November, 2003



Motivations

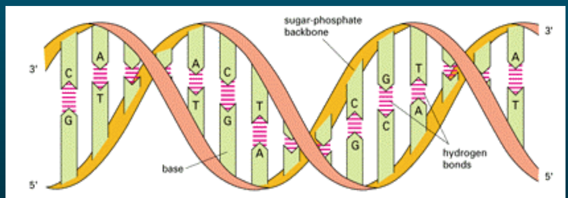
- Biological sequence search is bounded by I/O
  - Databases are exploding in sizes
  - Growing at an exponential rate
  - Exceeding memory capacity and thrashing
  - Accelerating performance gap between processors and I/O
- Goals
  - I/O characterization of biological search
  - Faster search by using parallel I/O
  - Using inexpensive Linux clusters
  - An inexpensive alternative to SAN or high end RAID

Talk Overview

- Background: biological sequence search
- Parallel I/O implementation
- I/O characterizations
- Performance evaluation
- Conclusions

Biological Sequence Search

- Unraveling life mystery by screening databases
  - Retrieving similar sequences from existing databases is important.
  - With new sequences, biologists are eager to find similarities.
  - Used in AIDS and cancer studies



unknown sequences

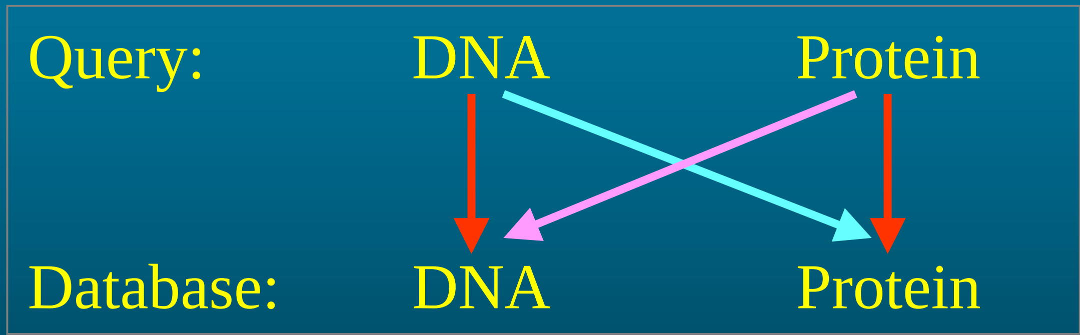
→

→ homologous sequences

Sequence Similarity
⇒ Structural Similarity
⇒ Functional Similarity

BLAST: Basic Local Alignment Search Tool

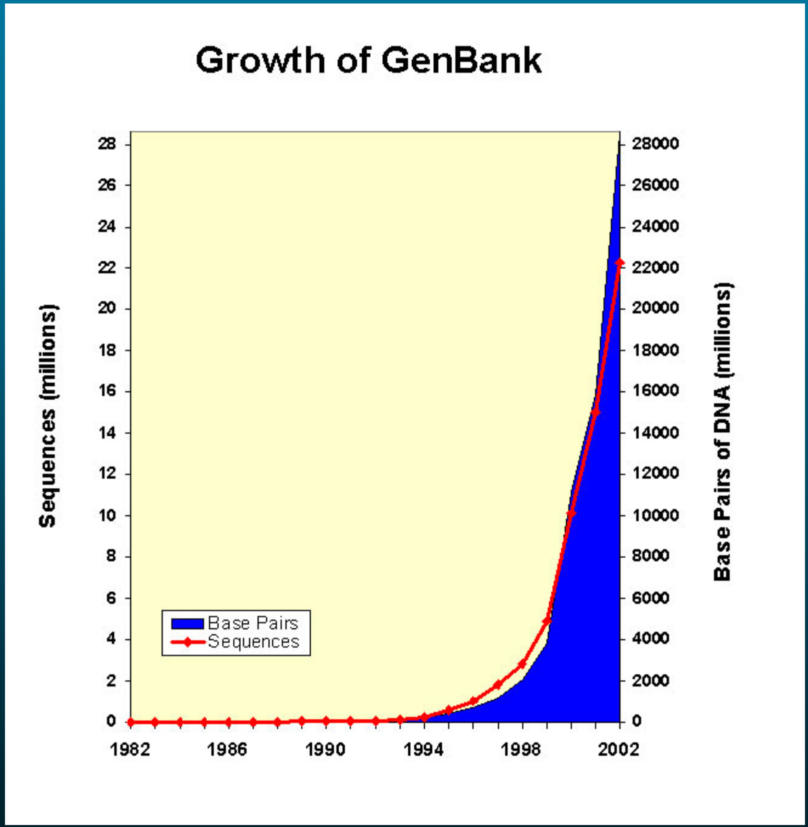
- BLAST
  - Most widely used similarity search tool
  - A family of sequence database search algorithms



- Parallel BLAST
  - Sequence comparison is embarrassingly parallel
  - Divide and conquer
    1) Replicate database but split query sequence
       Examples: HT-BLAST, cBLAST, U. Iowa BLAST, TurboBLAST
    2) Split database but replicate query sequence
       Examples: mpiBlast, TuboBlast, wuBLAST

Large Biology Databases

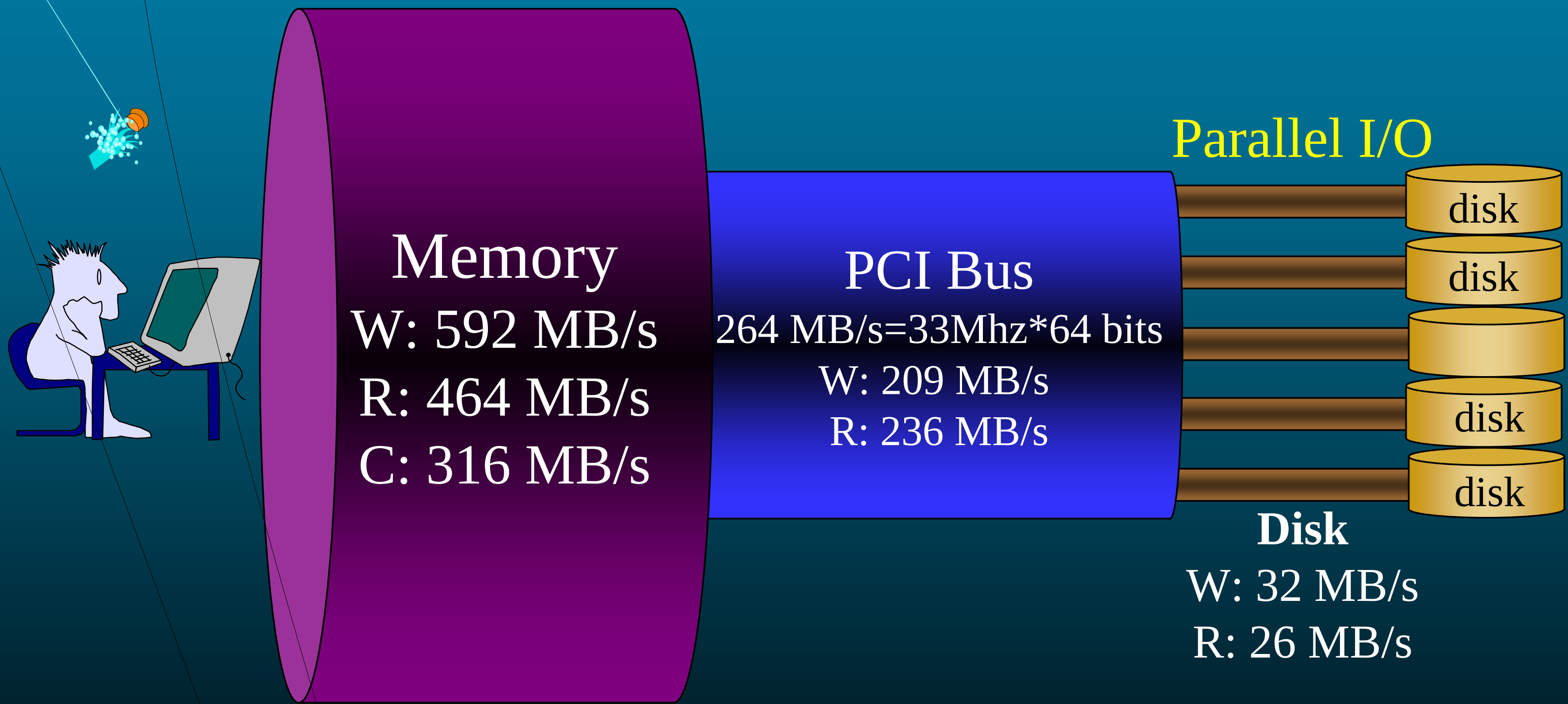
- Challenge: databases are large.
  - Public Domain
    - NCBI GenBank
      26 million sequences
      Largest database 8GBytes
    - EBI
      32 million sequences
      120 databases
  - Commercial Domain
    - Celera Genomics
    - DoubleTwist
    - LabBook
    - Incyte Genomics



Doubles every 13 months

Alleviate I/O Bottleneck in Clusters

The following data is measured on the PrairieFire Cluster at University of Nebraska:



Overview of CEFT-PVFS

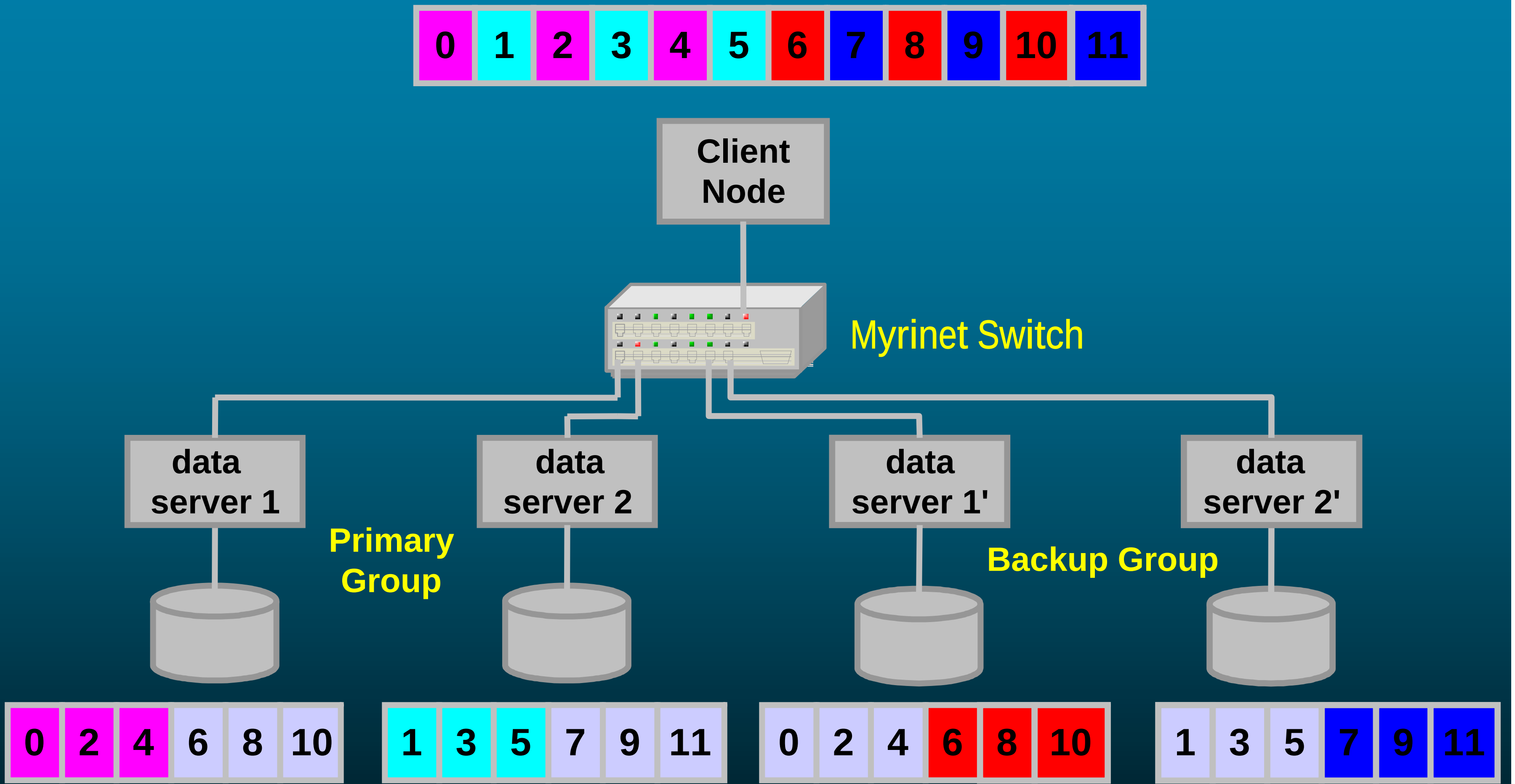
- Motivations
  - High throughput: exploit parallelism
  - High reliability:

    Assume MTTF of a disk is 5 years,

    MTTF of PVFS = MTTF of 1 node ÷ 128 = 2 weeks

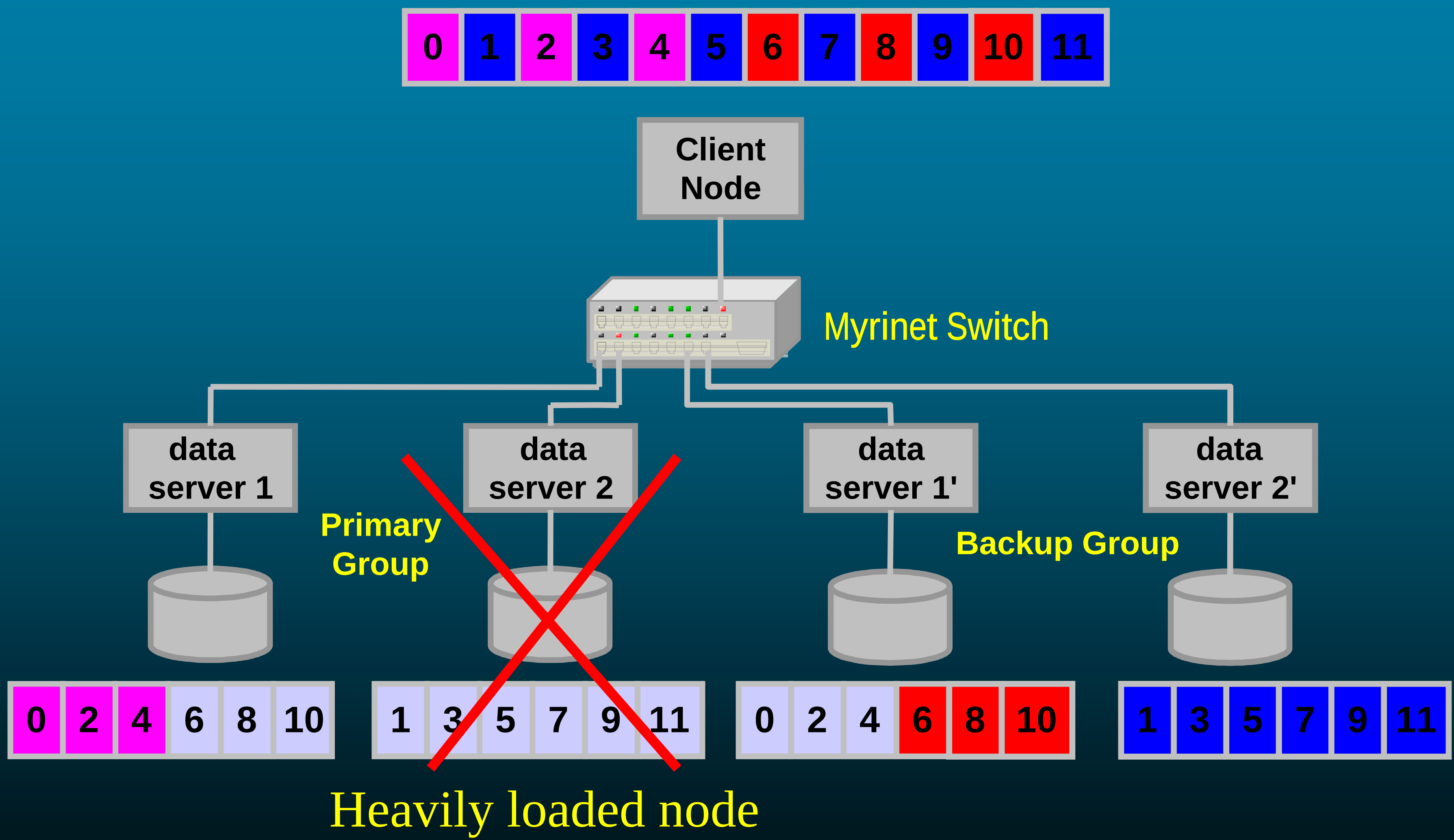
- Key features
  - Extension of PVFS
  - RAID-10 style fault tolerance
  - Free storage service, without any additional hardware costs
  - Pipelined duplication
  - “Lease” used to ensure consistency
  - Four write protocols to stride different balances between reliability and write throughput
  - Double parallelism for read operations
  - Skip hot spots for read

Double Read Parallelism in CEFT-PVFS



Interleaved data from both groups are read simultaneously to boost the peak read performance.

Skip Heavily Loaded Server



Parallel BLAST over Parallel I/O

- Avoid changing the search algorithms
- Minor modification to I/O interfaces of NCBI BLAST

  For example:

  read(myfile, buffer, 10)

  ⇒ pvfs_read(myfile, buffer, 10)

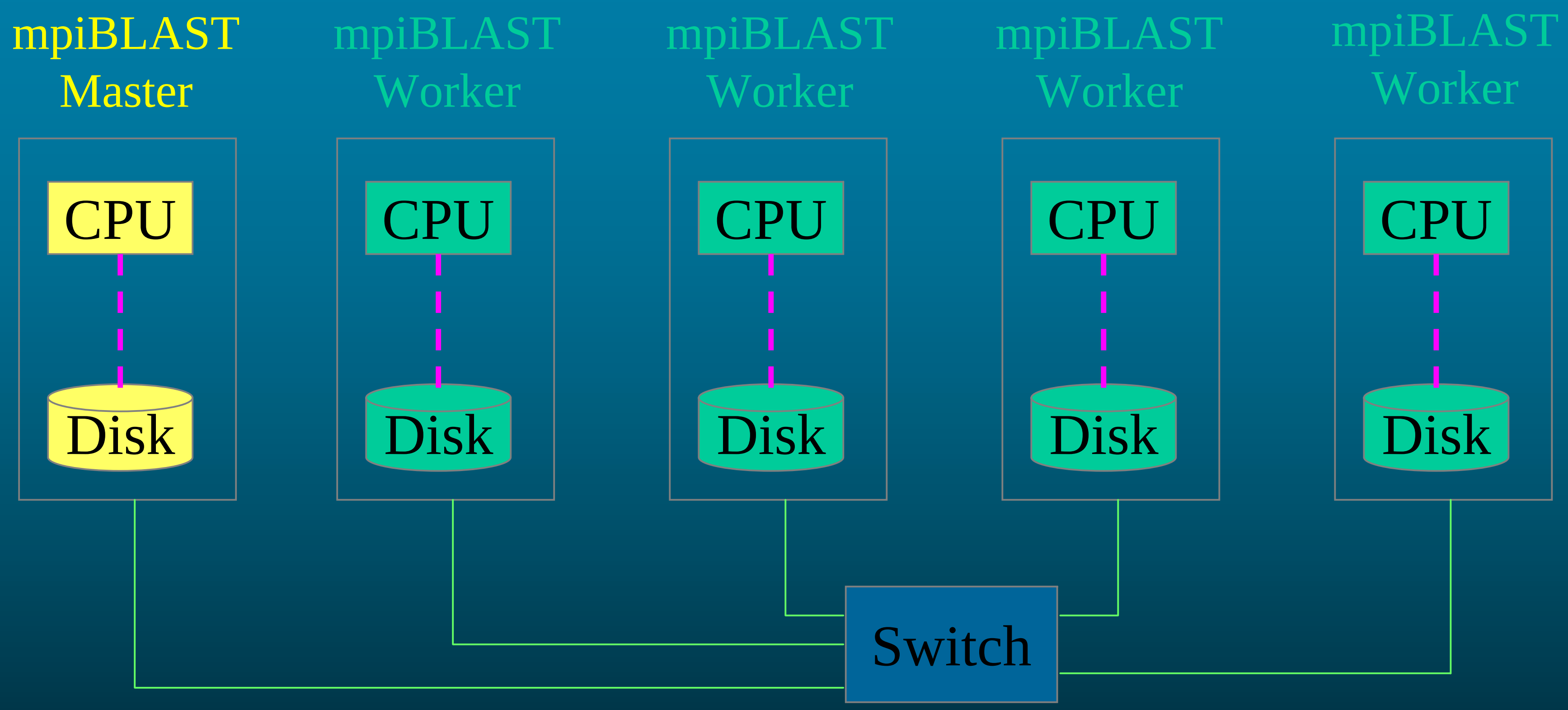
  ⇒ ceft_pvfs_read(myfile, buffer, 10)

- Arbitrarily choose mpiBLAST
- This scheme is general to any parallel blasts as long as they use NCBI BLAST

  (This is true in most case. ☺)

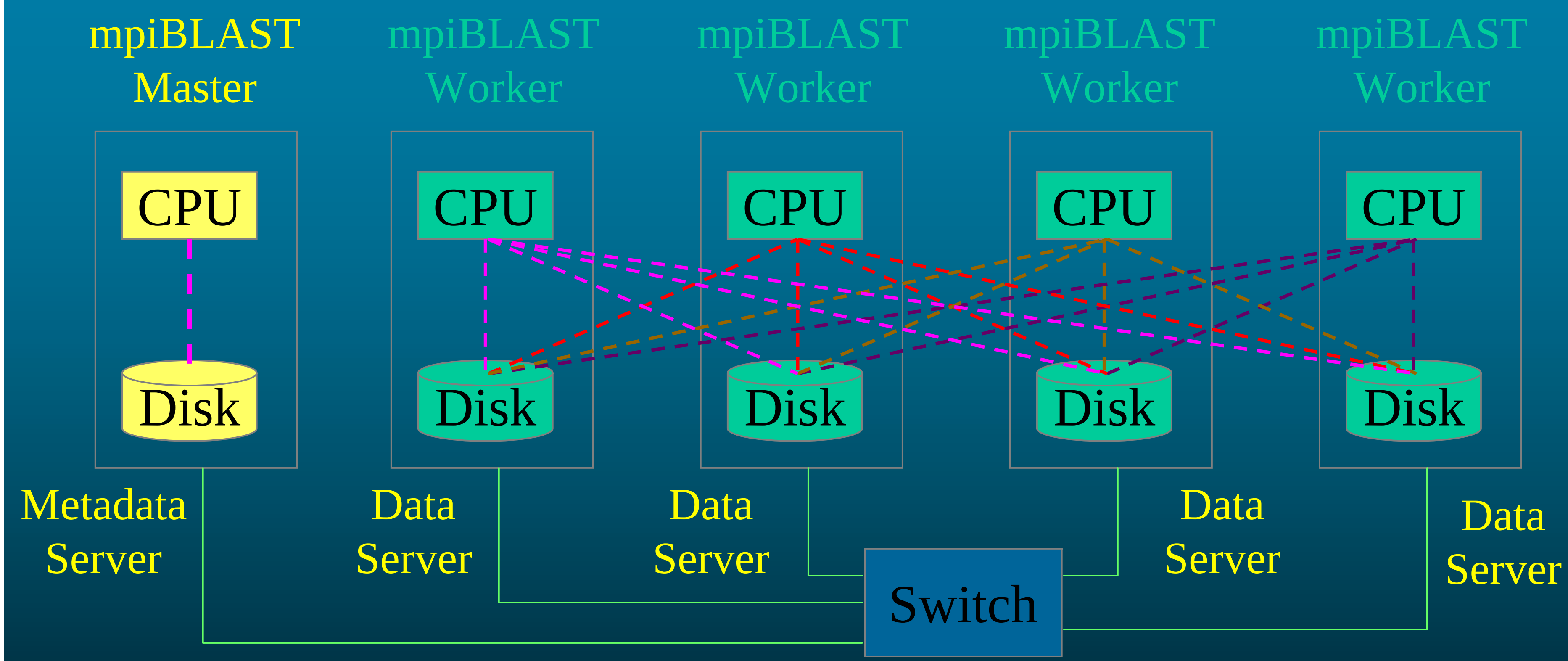
Software stack

Original mpiBLAST



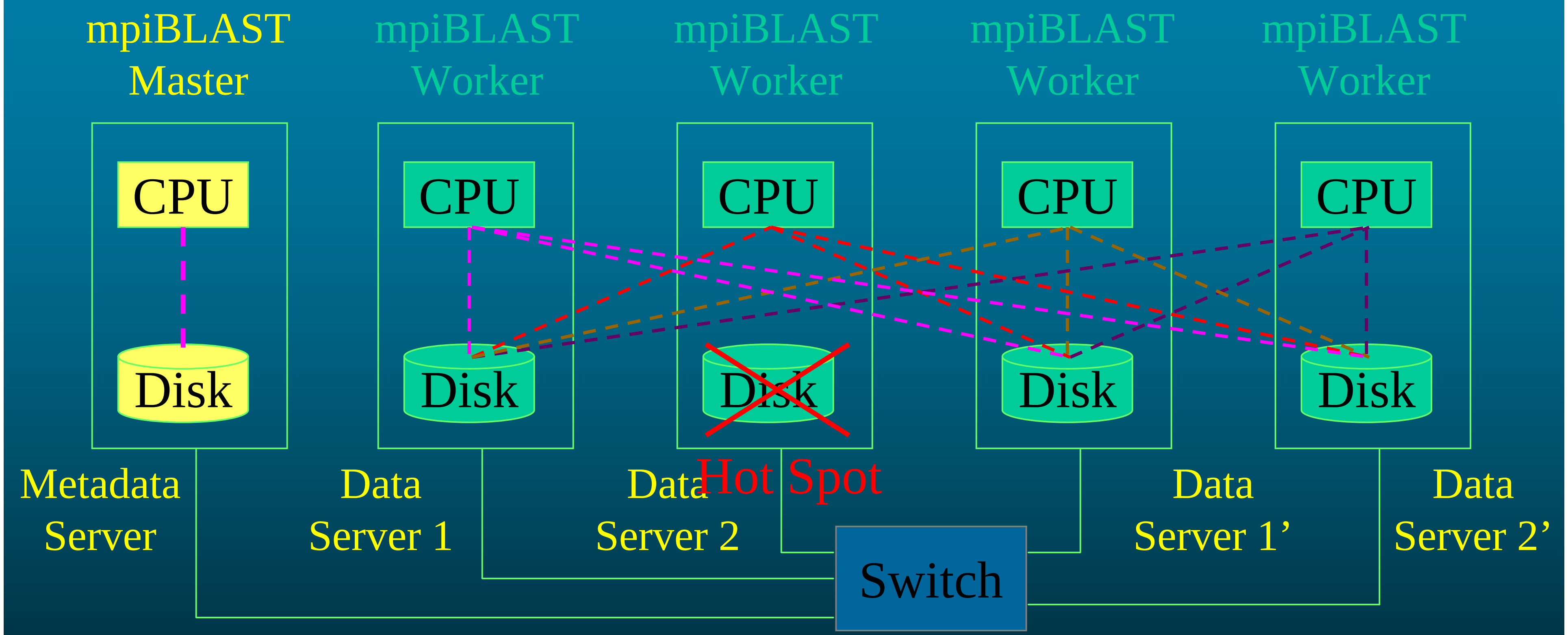
- **Master splits database into segments**
- **Master assigns and copies segments to workers**
- **Worker searches its own disk using entire query**

mpiBLAST over PVFS/CEFT-PVFS



- **Fair comparisons: Overlap servers and mpiBLASTs**
- **Database is segmented and then striped over data servers;**
- **Parallel I/O services for each mpiBLAST worker.**

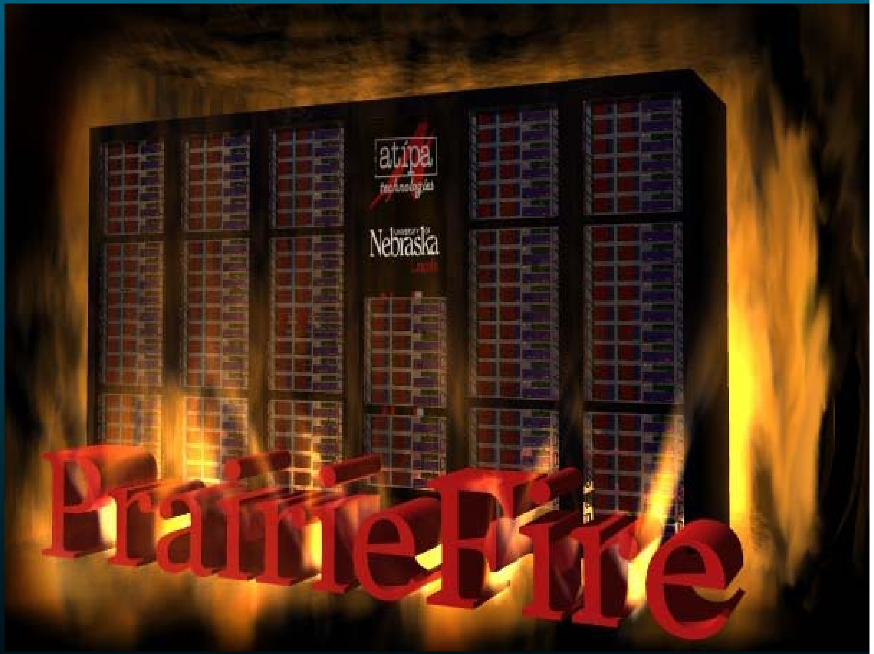
mpiBLAST over CEFT-PVFS with one hot spot skipped



- **Double the degree of parallelism for reads**
- **Hot spot may exist since servers are not dedicated**
- **Exploit redundancy to avoid performance degradation**

Performance Measurement Environments

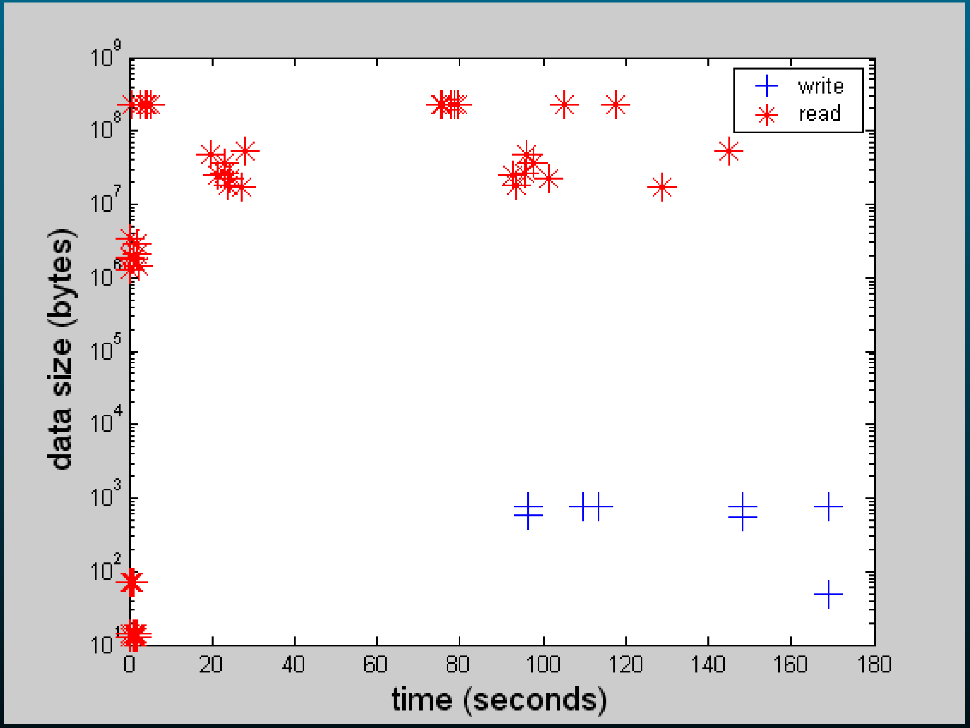
- PrairieFire Cluster at UNL
  - Peak performance: 716 Gflops
  - 128 IDEs, 2.6 TeraBytes total capacity
  - 128 nodes, 256 processors
  - Myrinet and Fast Ethernet
  - Linux 2.4 and GM-1.5.1



- Performance
  - TCP/IP: 112 MB/s
  - Disk writes: 32 MB/s
  - Disk reads: 26 MB/s

I/O Access Patterns

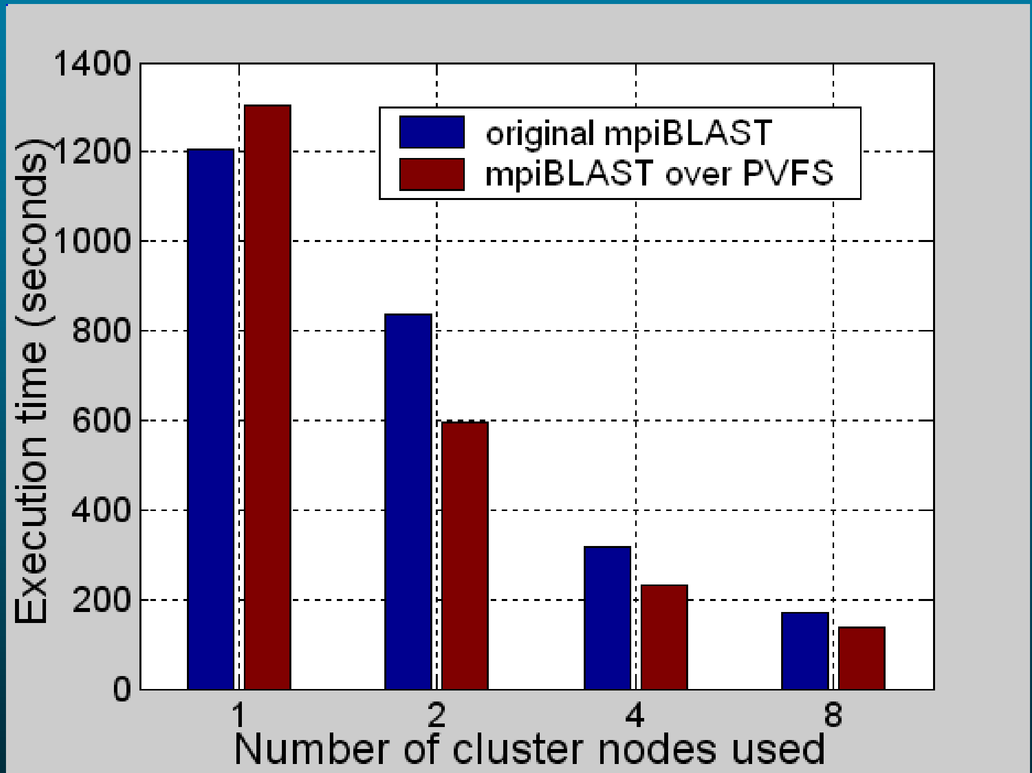
- Database: nt
  - largest at NCBI web
  - 1.76 million sequences, size 2.7 GBytes
- Query sequence: subset of ecoli.nt (568 bytes)
  - Statistics shows typical query length within 300-600 bytes
- I/O characteristics
  - 144 I/O operations,
  - 128 **reads** (89%), size: 13B ~ 220MB, mean 31.29 MB
  - 16 **writes** (11%), size: 50B ~ 779B, mean 690B



8 mpiBLAST workers

mpiBLAST *vs* mpiBLAST-over-PVFS

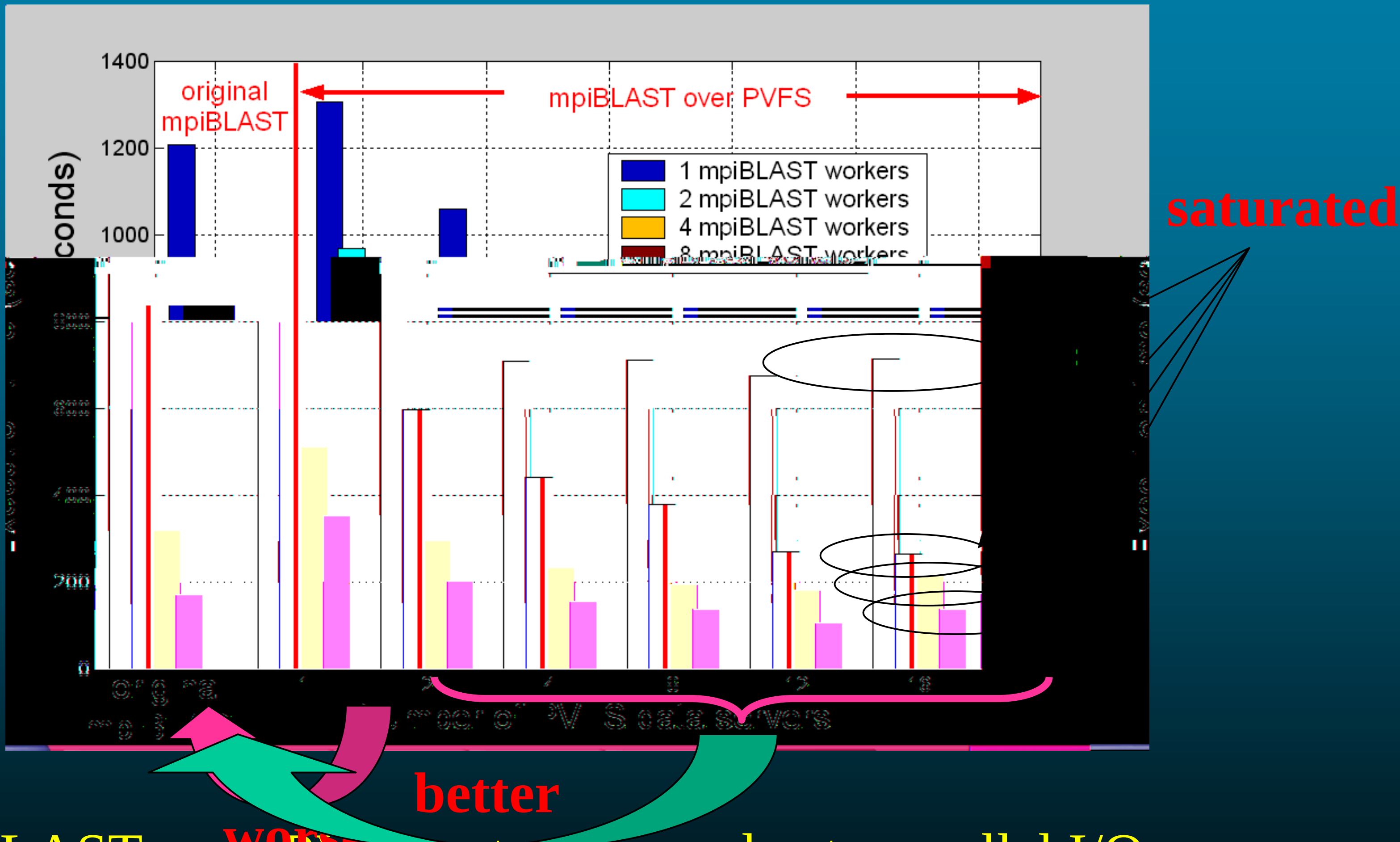
Comparison: assuming the SAME amount of resources



- With 1 node, mpiBLAST-over-PVFS performs worse.
- Parallel I/O benefits the performance.
- Gains of PVFS become smaller as node number increases.

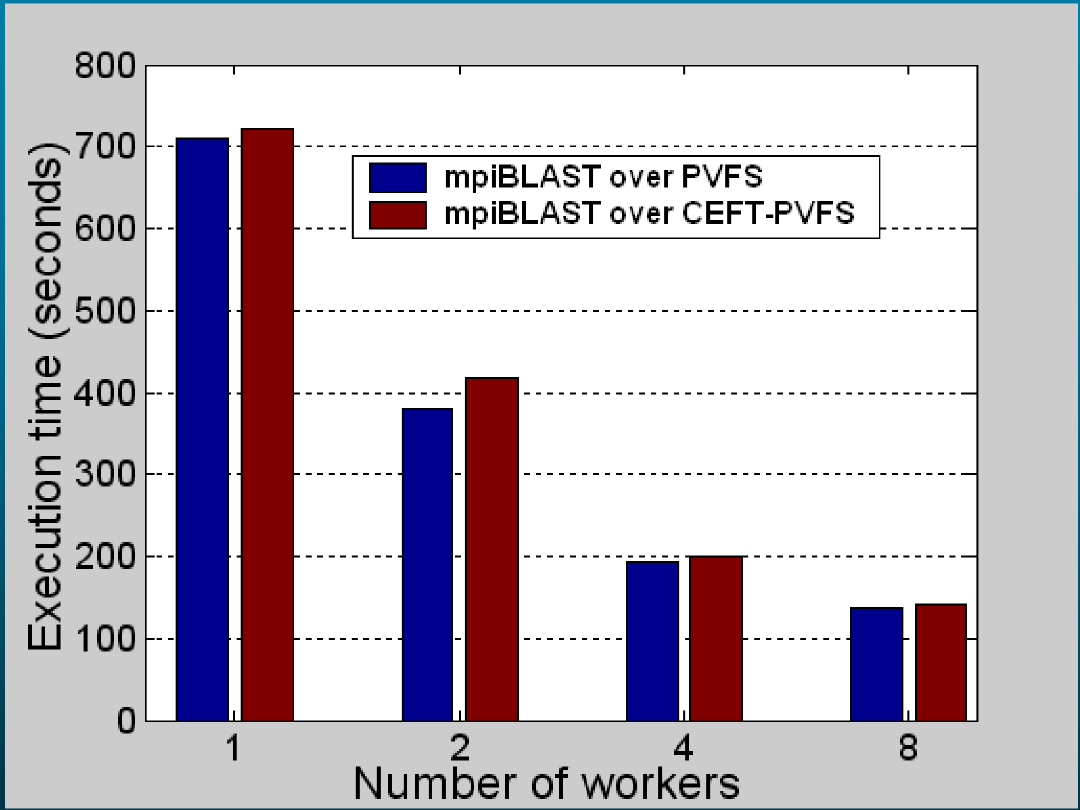
mpiBLAST *vs* mpiBLAST-over-PVFS (cont)

Comparisons: assuming **DIFFERENT** amount of resources



- mpiBLAST-over-PVFS outperforms due to parallel I/O.
- Gain from parallel I/O saturates eventually. (*Amdahl's Law*)
- As database scales up, grain becomes more significant.

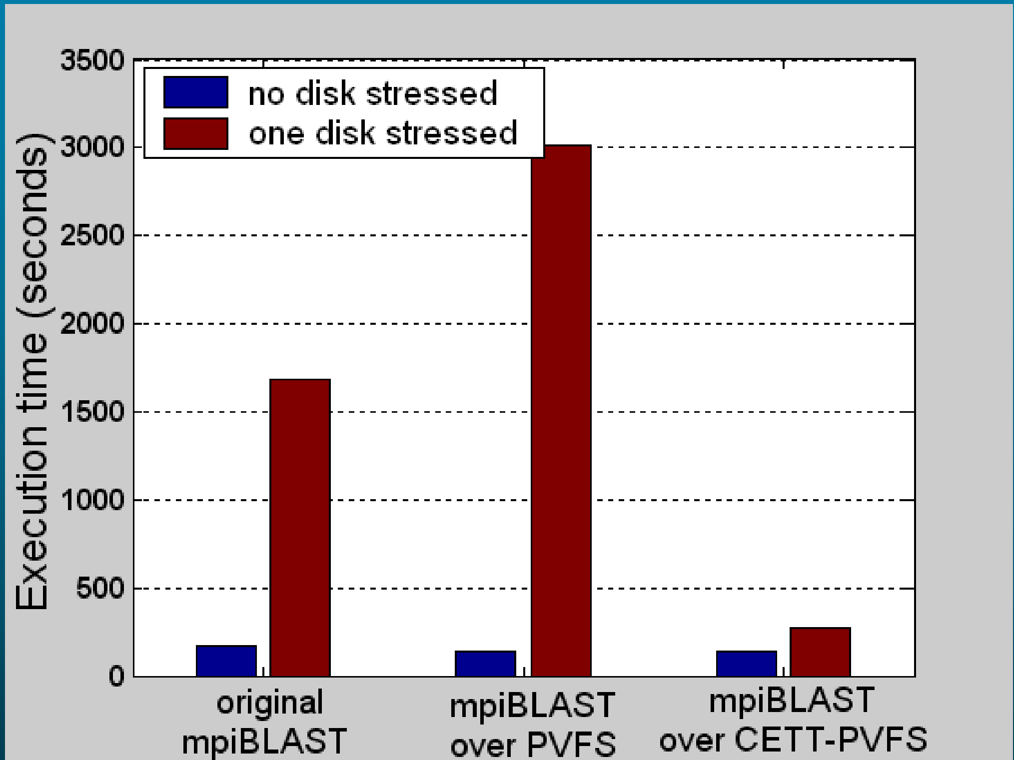
mpiBLAST-over-PVFS *vs.* over-CEFT-PVFS



PVFS: 8 data servers;
CEFT-PVFS: 4 mirrors 4.

- mpiBLAST is read-dominated
- Same degree of parallelism in PVFS and CEFT-PVFS for read
- Overhead in CEFT-PVFS is negligible

mpiBLAST-over-PVFS *vs* over-CEFT-PVFS




```
M = allocate(1 MBytes);
create a file named F;
while (true){
        if  ( size(F) > 2 GB ) {
                Truncate F to zero byte;
        else{
                Synchronously append M to F;
        }
}
```

disk stress program

- Configuration: PVFS 8 servers, CEFT-PVFS: 4 mirrors 4
- Stress disk on one data server by repeatedly appending data.
- While the original mpiBLAST and one over PVFS degraded by **10** and **21** folds, our approach degraded only by a factor of **2**.

Conclusions

- A higher degree of I/O parallelism may not lead to better performance.
- Doubling the degree of I/O parallelism for read operations provides CEFT-PVFS with a comparable read performance with PVFS.
- Load balance in CEFT-PVFS is important and effective.

Thank you

Question?

Abacus Distributd Storage Lab
University of nebraska - Lincoln

http://rcf.unl.edu/~abacus

Abacus Distributed Storage Lab

A Case Study of Parallel I/O for Biological Sequence Search on Linux Clusters

Yifeng Zhu, Hong Jiang, Xiao Qin, David Swanson

Department of Computer Science and Engineering

University of Nebraska – Lincoln

November, 2003