Effective File-I/O Bandwidth Benchmark (b_eff_io) and Other I/O Benchmarks

Rolf Rabenseifner
High-Performance Computing-Center Stuttgart (HLRS), University of Stuttgart,
rabenseifner@hlrs.de *www.hlrs.de/people/rabenseifner*

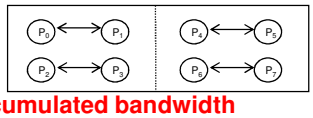
Invited Talk in the Lecture
"Hochleistungs-Eingabe/Ausgabe-Systeme"
Prof. Dr. habil Thomas Ludwig, Parallel and Distributed Systems,
Institute for Computer Science, University of Heidelberg
Jan 30, 2007

Outline

- Goals
- Comments on available benchmarking
- Definition of the b_eff and b_eff_io benchmarks
- Results
- Summary (b_eff_io)
- Some other I/O benchmarks
- Further information
- Questions / Lessons learned

Limits of some benchmarks

- Ping-Pong
  - is not a parallel benchmark
  - it is just a 2-processor-benchmark
  - buy 1000 dual-processor-PCs without any network
    --> **you will see perfect ping-pong bandwidth**
- accumulated bandwidth
  - buy 1000 dual...
    with a slow Ethernet
    --> **you will see perfect accumulated bandwidth**



- Bi-section bandwidth := minimum of accumulated bandwidth over all possible bi-sectioning
- Maximum I/O bandwidth
  - your application should **never** write a
    - small or medium-size package
    - and with size != 2**n

Goals for Communication and File-I/O Benchmarking

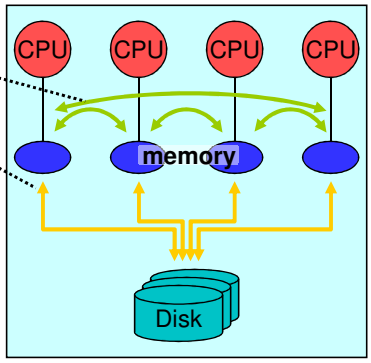
- Measure the time needed for exchange of information between
  - processes themselves, and
  - processes and disk
- Model the message passing or I/O patterns of real applications
- Provide a number for quick comparison of different systems
- Can't just measure simple send/receive or one I/O access:
  - Clock resolution
  - I/O caching
- So, traditional approach is to measure loops over specific patterns and quote e.g.,
  - 1) Ping-Pong Bandwidth
  - 2) Bi-Section Bandwidth
  - 3) Maximum I/O Bandwidth

Effective Communication & I/O Bandwidth Benchmarks

Goals

- Parallel **Communication** Benchmark
- Parallel **File-I/O** Benchmark

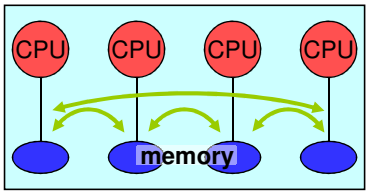
  each process is involved!
- Detailed insight
  - bandwidth experiments of several
    - I/O or communication patterns
    - chunk or message sizes
- One characteristic value
  - based on experiments above
  - averaging
- Appropriate execution time for rapid benchmarking



b_eff

the effective communication bandwidth benchmark

Definition of the Effective Communication Bandwidth Benchmark: b_eff



- www.hlrs.de/mpi/b_eff/
- **Authors: Karl Solchenbach, Hans-Joachim Plum, and Gero Ritzenhoefer (Pallas), Rolf Rabenseifner (HLRS)**

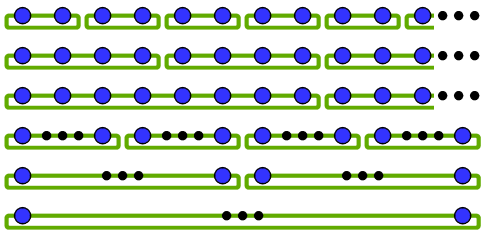
- 6 ring patterns
- 30 random patterns
- 13 additional patterns
- 21 message sizes
- 3 communication methods
- 3 times repeated

(6+30+13) x 21 x 3 x 3 = **9261 experiments**

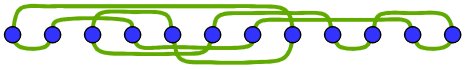
- automatically controlled measurement-loop length, i.e., time driven approach
- 5 - 20 msec / experiment —> benchmark completes in a **few minutes**

Definition of b_eff — communication patterns and sizes

- **6 ring patterns**
  - ring size = 2
  - 4
  - 8
  - max(#PE/4, 16)
  - max(#PE/2, 32)
  - #PE



- **30 random patterns**



- **21 message sizes**
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 byte, 1kB, 2kB, (12 sizes)
  - 9 logarithmic equidistant sizes: 4kB, ..., L_{max}= memory per PE / 128

Definition of b_eff — averaging

One characteristic accumulated communication bandwidth number
:= average bandwidth on several communication patterns
average on different message sizes
maximum over different MPI programming methods

$$b\_eff = logavg(\; logavg_{\mathbf{ring}pat} \; (avg_{\mathbf{L}}(max_{\mathbf{method}}(max_{\mathbf{rep}}(b_{\mathbf{pat,L,method,rep}})))) \; , \; logavg_{\mathbf{random}pat} \; (avg_{\mathbf{L}}(max_{\mathbf{method}}(max_{\mathbf{rep}}(b_{\mathbf{pat,L,method,rep}})))) \;)$$

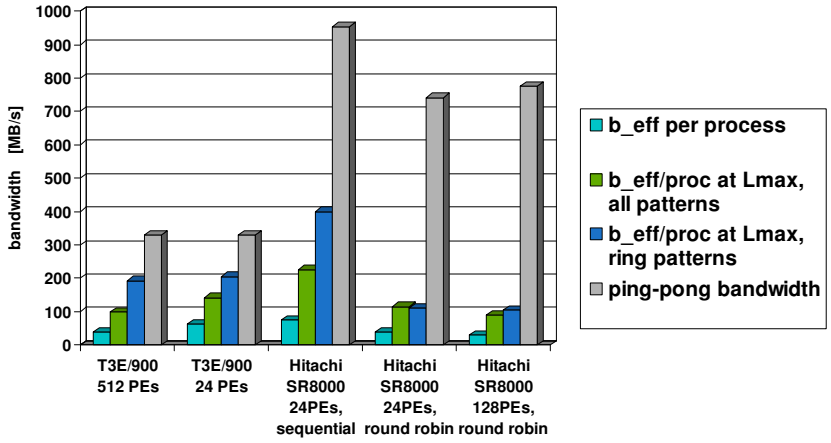
with

- $b_{\mathbf{pat,L,method,rep}}$ = accumulated bandwidth of each experiment
  over all processes
- methods: MPI_Sendrecv, MPI_Alltoallv, and nonblocking Irecv&Isend&Waitall
- pat & L: patterns and message sizes, see previous slide
- rep: repetition number = 1..3
- avg: arithmetic mean
- logavg: geometric mean

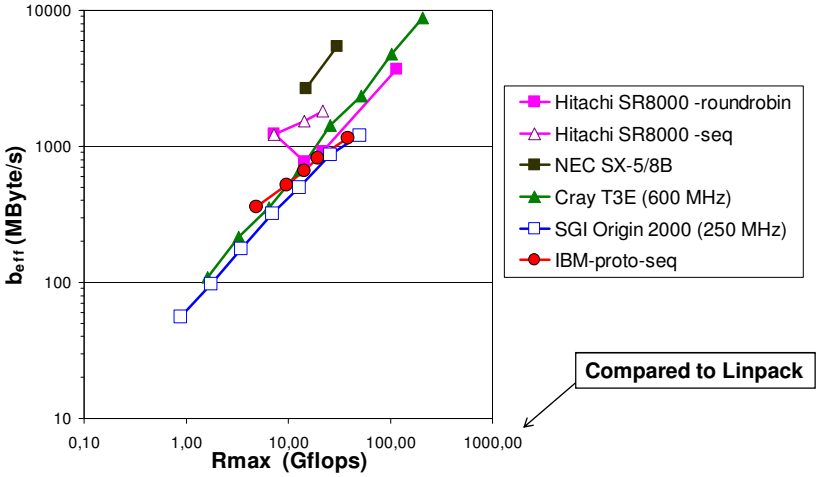
Features of Effective Bandwidth benchmark

- Based on MPI, source code is available
- Measures total architecture, not only point-to-point
- Checks performance of architecture and not the quality of the MPI implementation
- Suited for MPP-architectures and clusters
- Runs on any number of processors
- Results are easy to understand
- Generates a single number b_{eff} (like LINPACK R_{max})
- Aggregate bandwidth of the total system

b_eff per process and ping-pong bandwidth



B_eff Scaling



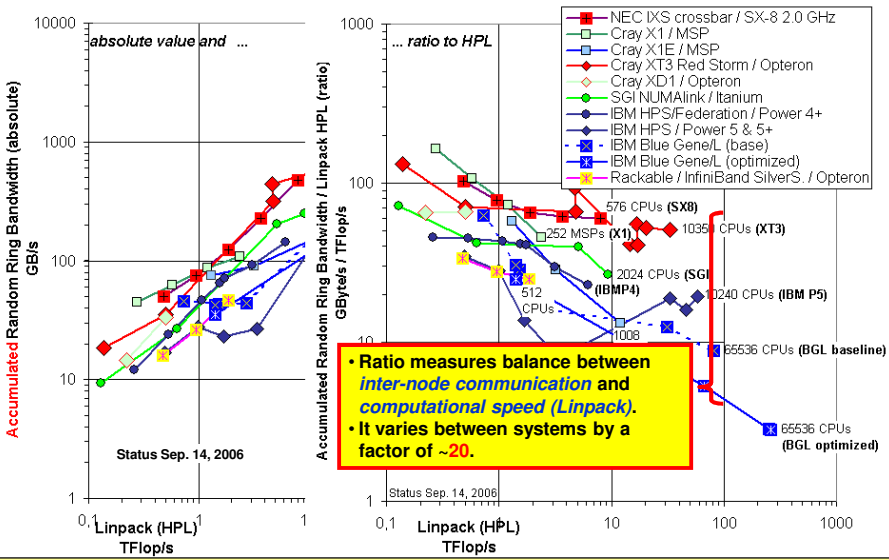
HPC Challenge Benchmark Suite

- High Performance Computing Challenge Benchmark Suite
- One benchmark is the natural & random ring bandwidth & latency benchmark
  - Using the b_eff ring technology
  - 2,000,000 byte messages for bandwidth
  - 8 byte messages for latency

Rolf Rabenseifner, Sunil R. Tiyyagurra, Matthias Müller: **Network Bandwidth Measurements and Ratio Analysis with the HPC Challenge Benchmark Suite (HPCC).** In Recent Advances in Parallel Virtual Machine and Message Passing Interface, Beniamino Di Martino, Dieter Kranzlmüller, and Jack Dongarra (Eds.), Proceedings of the 12th European PVM/MPI Users' Group Meeting, EuroPVM/MPI 2005, Sep. 18-21, Sorrento, Italy, LNCS 3666, pp 368-378, Springer, 2005.

Subhash Saini, Robert Ciotti,Brian T. N. Gunney, Thomas E. Spelce, Alice Koniges, Don Dossa, Panagiotis Adamidis, Rolf Rabenseifner, Sunil R. Tiyyagura, Matthias Mueller, and Rod Fatoohi: **Performance Evaluation of Supercomputers using HPCC and IMB Benchmarks.** Will be published in the proceedings of the IPDPS 2006 Conference, the 20th IEEE International Parallel & Distributed Processing Symposium, Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2006), Rhodes Island, Greece, April 25-29, 2006.

Balance: Random Ring B/W and HPL



Piotr R. Luszczek, David H. Bailey, Jack J. Dongarra, Jeremy Kepner, Robert F. Lucas, Rolf Rabenseifner, and Daisuke Takahashi: **The HPC Challenge (HPCC) Benchmark Suite.** Half-day Tutorial No. S-12 at Super Computing 2006, SC06, Tampa, Florida, USA, Nov. 11 - 17, 2006.

b_eff_io

the
effective MPI-I/O bandwidth benchmark

What about an I/O Benchmark? Starting-Points:

- Application benchmarks
  - using real, I/O-intensive applications
- File system benchmarks
  - measuring several parameters
    around the most friendly disk-usage-pattern
- Hardware benchmarks
  - maximum bandwidth of the disk — special-benchmark

- Why a new benchmark for parallel I/O?
  - application / file system / hardware independent
  - but, average on possible application scenarios
  - portable
    ==> MPI-I/O based benchmark

Starting-Points — the I/O Parameter Space

- How to define and measure one characteristic I/O bandwidth value?
- The I/O parameter space — 20 orthogonal parameters:
  - **Application parameters:**
    - **(a) the size of contiguous chunks in the memory, (b) on disk, (c) ... (f)**
  - **Usage aspects:**
    - **(a) how many processes are used**
    - **(b) how many parallel processors and threads are used for each process.**
  - **I/O interface:**
    - **(a) Posix I/O buffered or (b) raw,**
    - **(c) special filesystem I/O of the vendor filesystem,**
    - **(d) MPI-I/O.**
  - **MPI-I/O aspects:**
    - **(a) access methods, i.e., first writing of a file, rewriting or reading, (b) ...**
    - **(c) coordination, i.e., collectively or noncollectively, (d) ... (f)**
  - **Filesystem parameters:**
    - **(a) which filesystem is used,**
    - **(b) how many nodes are used as I/O servers, (c) ... (f)**

Full list, see: Rolf Rabenseifner and Alice E. Koniges: **Effective Communication and File-I/O Bandwidth Benchmarks.** In J. Dongarra and Yiannis Cotronis (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, proceedings of the 8th European PVM/MPI Users' Group Meeting, EuroPVM/MPI 2001, Santorini, Greece, Sep. 23-26, LNCS 2131, pp 24-35.

Existing I/O Benchmarking Techniques

- An example of I/O benchmarking papers:

  "Performance of the IBM General Parallel File System,"
  Terry Jones, Alice Koniges, R. Kim Yates,
  Proceedings of the International Parallel and Distributed Processing Symposium, May 2000. Also available as UCRL JC135828

- many hours of dedicated benchmarking time is used
- characterizing a specific system
- not portable

Goals for b_eff_io

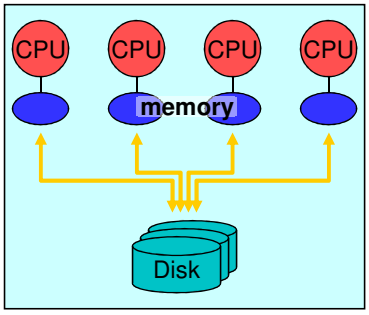
- portable
- characterizing parallel I/O capabilities of a system
- many patterns
- ... that can be optimized
- going beyond the caching capabilities
  --> measuring the real disk I/O

Rule: Balanced HPC systems should be able to write the total memory in 10 minutes to disk

==> **An I/O benchmark should not need hours!**
— 10 minutes may be enough to overrun any cache!

- time driven approach / automatically controlled repetition factors
- rapid benchmarking (30-60 min.)

Definition of the Effective File-I/O Bandwidth Benchmark: b_eff_io

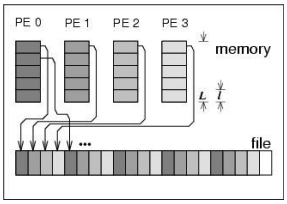


- 5 I/O patterns
- 7 chunk sizes
- 3 accesses
  (initial write, rewrite, read)
- 3 compute partition sizes
  (number of parallel benchmark processes)

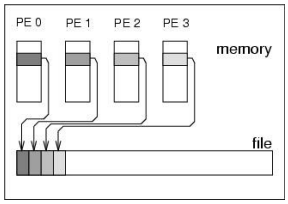
315 different measurements

- benchmark completes in **30-60 minutes**
- www.hlrs.de/mpi/b_eff_io/

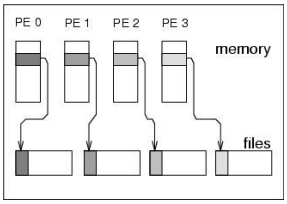
Definition of b_eff_io — the Pattern Types



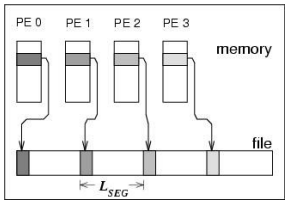
type 0: scattering *)



type 1: strided coll.



type 2: noncoll., separated



type 3 noncoll. / 4 coll. segmented

Pattern that can be optimized

Chunk sizes on disk:

- $\mathbf{L_{max}}$ = max (2MB, memory of one node/128) *)
- **wellformed**: 1MB, *) 32 kB, 1 kB,
- **non-wellformed**: 1MB+8B, *) 32 kB+8B, 1kB+8B

*) double weighted

Definition of b_eff_io

(Release 1.0)

b_eff_io := Maximum over all usage and filesystem parameters } manually

Average on write, rewrite, read
Average on five access pattern types
Average on several chunk size values*)
of measured bandwidth
} auto-matically, in time T=30 min.

*) defines the size of contiguous chunks written to disk and the contiguous chunk in memory written by each MPI call

Output of the b_eff_io benchmark program

- the b_eff_io value

weighted average bandwidth for **write** : **422.**388 **MB/s** on 64 processes
weighted average bandwidth for **rewrite** : **331.**833 **MB/s** on 64 processes
weighted average bandwidth for **read** : **473.**441 **MB/s** on 64 processes
Total amount of data written/read with each access method: **208558.**575 MBytes
= **39.8** percent of the total memory (**524288** MBytes)
b_eff_io of these measurements = 451.727 **MB/s**
on 64 processes with 2048 MByte/PE, scheduled time=30.0 Min,
on AIX frost018 3 4 006005094C00
total memory / **b_eff_io =** 524288 Mbytes / 451.727 MB/s **= 19.3 min**.

- detailed results
  - as ASCII table
  - one page with 3+5 plots
    - **all measurements sorted by access: write / rewrites / reads**
    - **and same sorted by pattern types: type-0 / type-1 / type-2 / type-3 / type-4**

Time-driven approach

- **b_eff**
  - for each message size:
    loop length is based on
    execution time of next smaller message size
  - starting loop length for each pattern and method
    = 300 (release <= 3.3)
    = based on a quick latency measurement with 10 iterations (rel.>3.3)
- **b_eff_io**
  - first write
    & pattern types 0-2 (**scatter collective, shared collective, separated files**):
    • writing until scheduled time is over for each pattern and chunk size
  - first write & pattern types 3+4 (**segmented file, collective and not**):
    • pre-calculated repeating factors,
    • based on measured execution times with pattern types 0-2
  - rewrite & read: same amount of data as with *"first write"*

Definition of b_eff_io — Bandwidth measurement

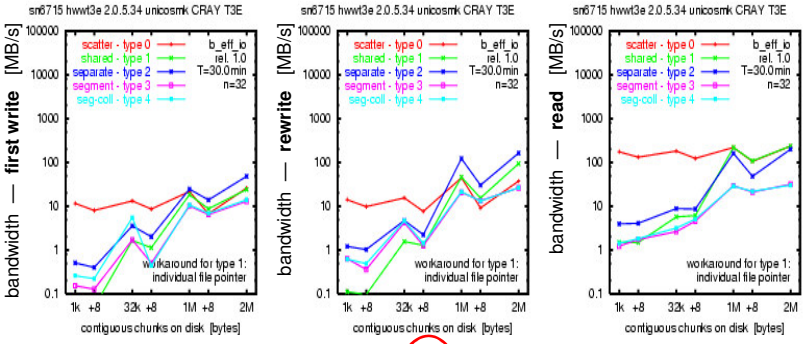
- **Bandwidth measurement**

```
MPI_Barrier()
start_time = MPI_Wtime()   at root only
repeat
   MPI_File_write() or MPI_File_read()
   MPI_Barrier()
   conti = (MPI_Wtime()- start_time) < time_unit
   MPI_Bcast(conti)
while conti
if (write access) MPI_File_sync()
MPI_Barrier()
end_time = MPI_Wtime()   at root only
bandwidth = (accumulated data size)
            / (end_time - start_time)
```


I/O Results — Comparing systems

- Cray T3E 900-512 at HLRS/RUS, Stuttgart
  - 512 processors
  - 10 striped Raid-disks, connected via GigaRing
  - mpt.1.3.0.2 with ROMIO, modified: using asynchronous I/O
    - **www.hlrs.de/mpi/mpi_t3e.html#StripedIO**
    - **www.hlrs.de/mpi/ufs_t3e/**
  - theoretical peak throughput = 300 MB/s
- IBM RS 6000/SP at LLNL, called “blue pacific”
  - 336 SMP nodes with each 4 processors
  - benchmark: using 1 processor per node
  - IBM General Parallel File System (GPFS) with 20 VSD I/O server
  - ROMIO
  - measured peak performance: 950 MB/s read, 690 MB/s write (on 128 nodes)
- NEC SX-5Be/32M2 at HLRS/RUS, Stuttgart
  - 2 SMP nodes with each 16 processors
  - benchmark only on one SMP node
  - SFS filesystem, 4MB block size
  - I/O requests less than 1 MB are cached on 2 GB filesystem-memory-cache

Detailed Results – Cray T3E-900/512 at HLRS *)



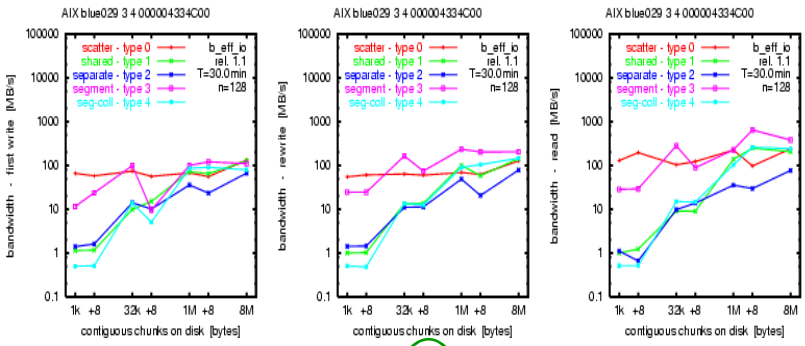
weighted average bandwidth for write: 12 MB/s on 32 processes (25%)
weighted average bandwidth for rewrite: 21 MB/s on 32 processes (25%)
weighted average bandwidth for read: 98 MB/s on 32 processes (50%)

b_eff_io of these measurements = 57 MB/s on 32 processes

Quite different !

*) with disk striping, patches at www.hlrs.de/mpi/mpi_t3e.html

Detailed Results – IBM SP "Blue Pacific" at LLNL

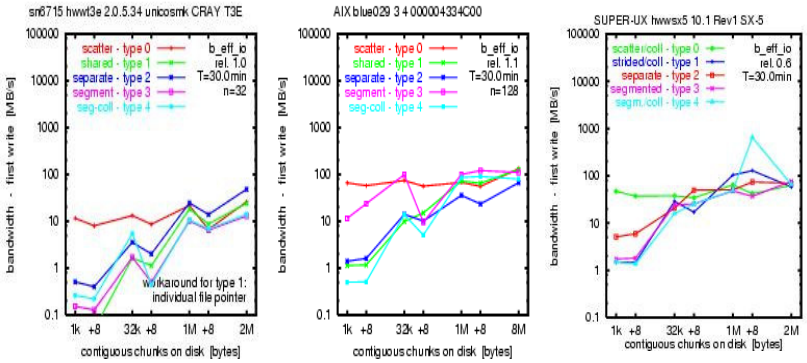


weighted average bandwidth for write: 53 MB/s on 128 processes (25%)
weighted average bandwidth for rewrite: 68 MB/s on 128 processes (25%)
weighted average bandwidth for read: 65 MB/s on 128 processes (50%)

b_eff_io of these measurements = 63 MB/s on 128 processes

Nearly the same

Results: "write" on Cray T3E, IBM SP, and NEC SX-5

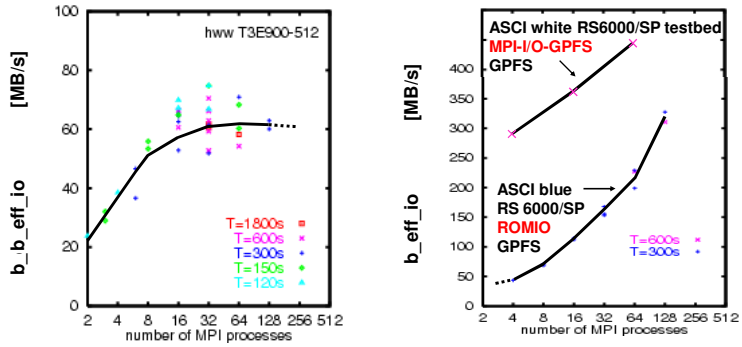


Cray T3E
32 Pes
T=30 min.
(b_eff_io=57 MB/s)

IBM SP
128 nodes
T=30 min.
(b_eff_io=63 MB/s)

NEC SX-5
4 nodes
T=30 min.
(b_eff_io=60 MB/s)

First Results — Comparing b_eff_io (#processes)



Cray T3E:
- reached already with 8 processors!
  => optimal for any load:
  many small jobs ... one large job

IBM SP:
- with optimized MPI-I/O-GPFS
  – high bandwidth for any load
- with ROMIO
  – I/O capability bound to application
  – full bandwidth only with many processes

First Results — Interpretation

- maximum bandwidth / partition sizes
- small influence of scheduled time T
- benchmarked platforms: MPI-I/O is optimal only for one pattern type
- but different optimal type on each platform
- non-wellformed data sizes: worse I/O bandwidth
- (re)write bandwidth << read bandwidth
- no chance to predict bandwidth for other patterns

Optimization with MPI-I/O Hints

- Hints (Info argument) can be given on
  - MPI_File_open (…, fh, info) when file is created
  - MPI_File_open (…, fh, info) in general
  - MPI_File_set_view (fh, …, info)
  - MPI_File_set_info (fh, info)
- Some hints are used only, e.g.,
  - when file is created, e.g.,
    - **Number of strip units**
    - Strip size
- Standardized hints and vendors' hints

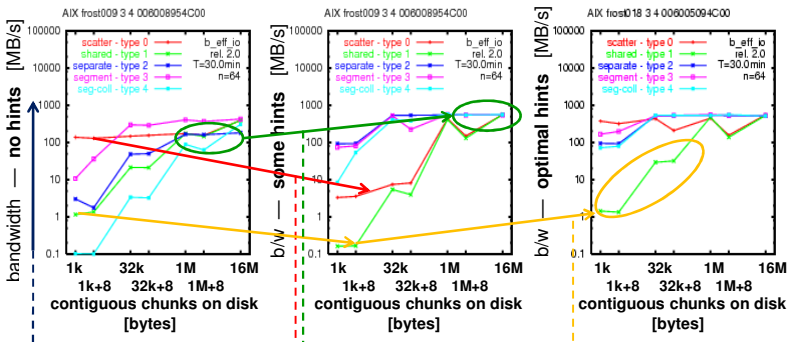
Optimization with b_eff_io

- Hints can be given on a file
  - Some hints on IBM GPFS system:
  - **IBM_large_block_io=false & IBM_io_buffer_size=32MB (instead of 8MB) for "*scatter*" and "*shared*" pattern type with chunk sizes <= 32k+8**

```
# Hints for using IBM_largeblock_io and IBM_io_buffer_size on LLNL frost
 all all all all IBM_largeblock_io true
 all all scatter 1024 IBM_largeblock_io false
 all all scatter 1032 IBM_largeblock_io false
 all all scatter 32768 IBM_largeblock_io false
 all all scatter 32776 IBM_largeblock_io false
 all all shared 1024 IBM_largeblock_io false
 all all shared 1032 IBM_largeblock_io false
 all all shared 32768 IBM_largeblock_io false
 all all shared 32776 IBM_largeblock_io false
# 32 MB = 64 GPFS block size, with 1 GPFS block = 512KB
 all all scatter 1024 IBM_io_buffer_size 32MB
 all all scatter 1032 IBM_io_buffer_size 32MB
 all all scatter 32768 IBM_io_buffer_size 32MB
 all all scatter 32776 IBM_io_buffer_size 32MB
 all all shared 1024 IBM_io_buffer_size 32MB
 all all shared 1032 IBM_io_buffer_size 32MB
 all all shared 32768 IBM_io_buffer_size 32MB
 all all shared 32776 IBM_io_buffer_size 32MB
```

Maybe not used !?

Results on ASCI White testbed with different hints:
First write



Logarithmic scale: 4 orders of magnitude !

Things may go worse !

Results may be better !

Some pattern may be resistant to any optimization

Optimization with b_eff_io

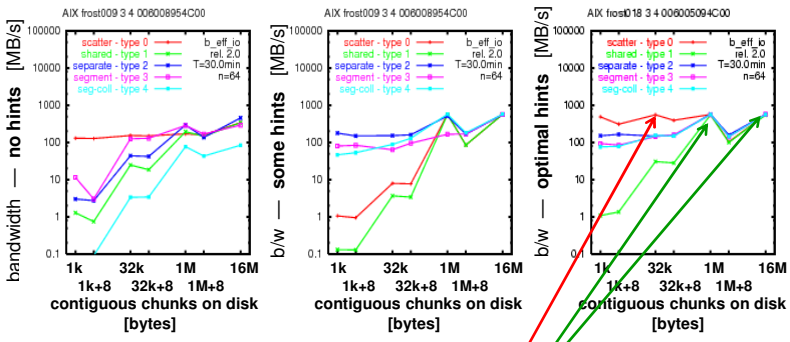
- Optimized hints: Now `IBM_largeblock_io switchable` and `2MB`

```
# Hints for using IBM_largeblock_io and IBM_io_buffer_size on LLNL frost
# - the following line overwrites the default:
    all all all          all IBM_largeblock_io true
# - overwriting this for opening the file:
    all all all     on_open IBM_largeblock_io switchable
    all all scatter    1024 IBM_largeblock_io false
    all all scatter    1032 IBM_largeblock_io false
    all all scatter   32768 IBM_largeblock_io false
    all all scatter   32776 IBM_largeblock_io false
    all all shared     1024 IBM_largeblock_io false
    all all shared     1032 IBM_largeblock_io false
    all all shared    32768 IBM_largeblock_io false
    all all shared    32776 IBM_largeblock_io false
# 2 MB = 4 GPFS block size, with 1 GPFS block = 512KB
    all all all     on_open IBM_io_buffer_size switchable
    all all scatter    1024 IBM_io_buffer_size 2MB
    all all scatter    1032 IBM_io_buffer_size 2MB
    all all scatter   32768 IBM_io_buffer_size 2MB
    all all scatter   32776 IBM_io_buffer_size 2MB
    all all shared     1024 IBM_io_buffer_size 2MB
    all all shared     1032 IBM_io_buffer_size 2MB
    all all shared    32768 IBM_io_buffer_size 2MB
    all all shared    32776 IBM_io_buffer_size 2MB
```

Hard to know / guess --- even for the specialists !!

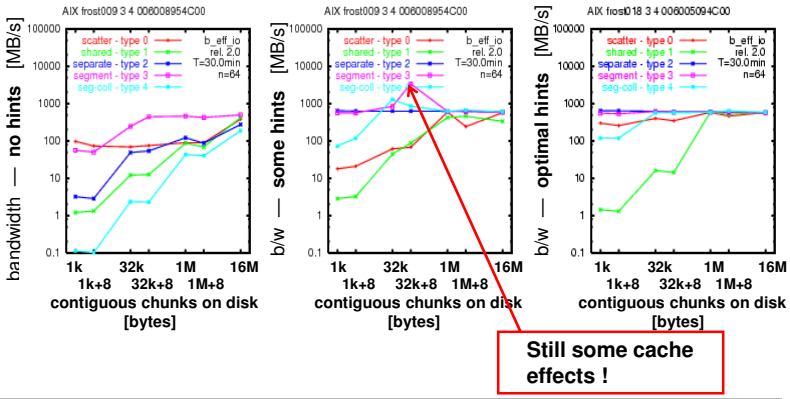
all/#processes | all/write/rewrite/read | all/pattern type | all/size | hint | value

Results on ASCI White testbed with different hints: Rewrite



Optimized rewrite below 200 MB/s for all pattern and chunk sizes except scatter type 0 and 1 MB and 16 MB

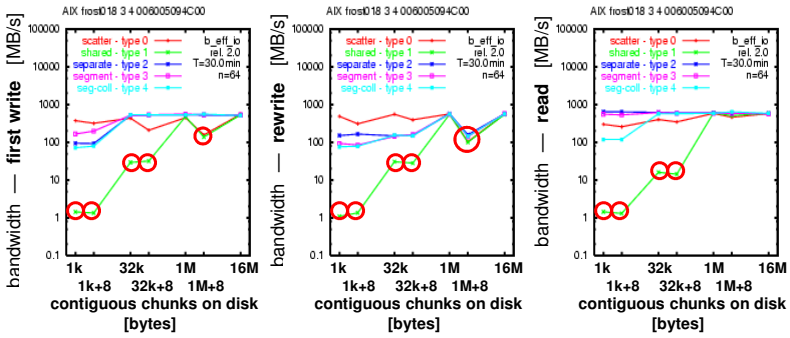
Results on ASCI White testbed with different hints: Read



Still some cache effects !

- **Optimized read bandwidth for several patterns > 400 MB/s**

Results on ASCI White testbed with optimal hints



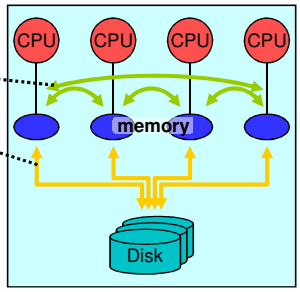
- **each pattern may be optimized with hints** *)
  - **here: IBM_large_block_io=false & IBM_io_buffer_size=2MB (instead of 8MB) for "*scatter*" and "*shared*" pattern type with chunk sizes <= 32k+8**
  - **86 from 105 patterns quite fast ! 19 candidates for further optimization**

*) Rolf Rabenseifner, Alice E. Koniges, Jean-Pierre Prost, and Richard Hedges: **The Parallel Effective I/O Bandwidth Benchmark: b_eff_io**. In Christophe Cerin and Hai Jin (Eds.), **Parallel I/O for Cluster Computing**, Chap. 4. (pp 107-132), Kogan Page Ltd., Feb. 2004, ISBN 1-903996-50-3.

Summary

Two parallel benchmarks:

- b_eff: Communication
- b_eff_io: File-I/O
- Detailed insight
  - many patterns, chunk sizes, ...
- One characteristic value
  - averaging
- Appropriate execution time for rapid benchmarking
  - time driven approach
  - b_eff: 3–5 min.
  - b_eff_io: 30–60 min.
- Optimization with hints is possible / necessary



Further information:

www.hlrs.de/mpi/b_eff	**&**	**.../b_eff_io**
www.hlrs.de/mpi/mpi_t3e.html	**&**	**.../ufs_t3e**
www.hlrs.de/people/rabenseifner/publ/publications.html		

Further I/O benchmarks

Low level I/O Benchmarks:

- IO_Bench
- SPIOBENCH
- MPI Tile I/O Benchmark
- IOR Benchmark (Interleaved Or Random)
- MPI pio benchmark
- b_eff_io

Compact Application Benchmarks:

- Flash I/O
- HPCS SSCA #3

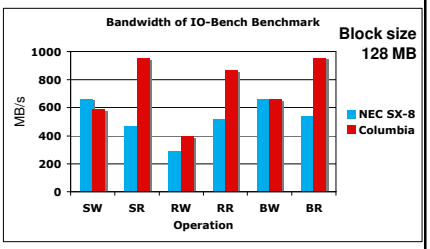
Platforms used for following benchmarks

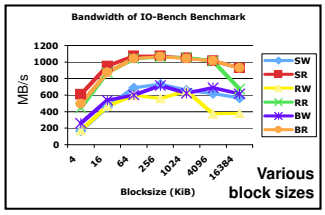
Characteristic	Columbia	NEC SX-8
System size	20 nodes	72 nodes
# Processors per node	512	8
Total number of processors	10,240	576
Peak performance (Tflop/s)	65. 536	9.216
Memory per processor / totally	2 GB / 20 TB	16 GB / 9 TB
Network	NUMALink4	IXS
System vendor	SGI	NEC
File system type	CXFS	GStorageFS
FC2 ports at node	8 ports per node	4 ports per node
FC2 ports at disks	8	72
I/O FC limit per node	1.6 GB/s	0.8 GB/s
I/O FC limit total	1.6 GB/s	14.4 GB/s
Used	Up to 1 node = 512 CPUs	Up to 32 nodes = 256 CPUs

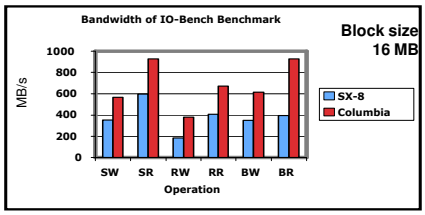
Details, see
Subhash Saini, Dale Talcott, Rajeev Thakur, Panagiotis Adamidis, Rolf Rabenseifner, Robert Ciotti:
Parallel I/O Performance Characterization of Columbia and NEC SX-8 Superclusters.
Will be published in the proceedings of the IPDPS 2007 Conference,
the 21th IEEE International Parallel & Distributed Processing Symposium, Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2007), Long Beach, California, USA, March 26-30, 2007.

IO_Bench

- Single CPU Benchmark
  - Sequential write and read (SW & SR)
  - Random write and read (RW & RR)
  - Backward write and read (BW &BR)
- Percentage of I/O peak (128MB block size)
  - Read: Columbia 54-60%, NEC 59-67%
  - Write: Columbia 25-41%, NEC 36-83%

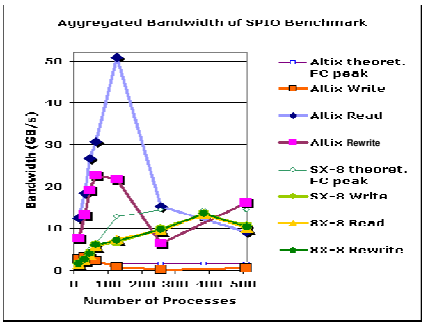


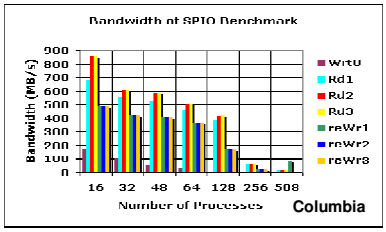


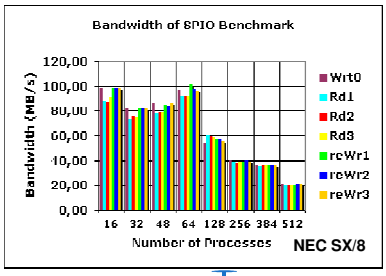


SPIOBENCH Benchmark

- Test scalability
- Aggregate I/O: 128 GB
- Columbia: Extreme b/w on memory cache
  - **Reported numbers are significant larger than physical peak**
- NEC SX-8: 53-96% of peak b/w on real disk

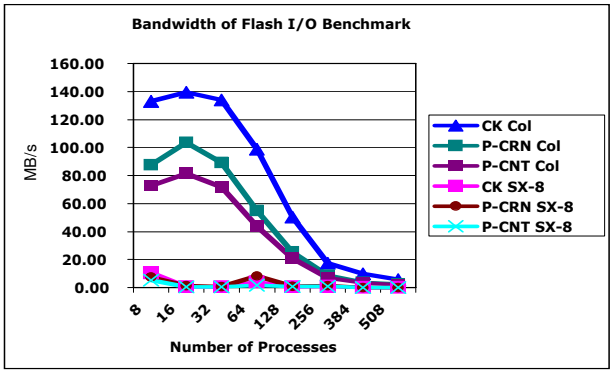






Flash I/O Benchmark

- For a standard grid of 8x8x8
- Uses HDF5 library
- 3 Output files:
  - Check point (CK)
  - Plot file for centered data (P-CRN)
  - Plot file for corner data (P-CNT)



Acknowledgements

Thanks to pallas.
They initiated this project with their bi-section based b_eff benchmark.

Parts of the work by Lawrence Livermore National Laboratory was performed under the auspices of the U.S. Department of Energy by the University of California under Contract W-7405-ENG-48, UCRL-VG-143637.

Thanks to all people who worked with me on this topic:
Alice E. Koniges, Karl Solchenbach, Jean-Pierre Prost, Richard Hedges, Subhash Saini, Dale Talcott, Rajeev Thakur, Panagiotis Adamidis, Robert Ciotti, Rolf Hempel, Thomas Bönisch, Luis Miguel Sanchez Garcia

Further information

Papers:	www.hlrs.de/people/rabenseifner/publ/publications.html
b_eff	www.hlrs.de/mpi/b_eff
b_eff_io	www.hlrs.de/mpi/b_eff_io
IO_Bench	see http://www.erdc.hpc.mil → HPCMO, IOBench; and also www.sdsc.edu/PMaC/Benchmark/iobench/
SPIOBENCH	see information at http://www.nsf.gov/pubs/2006/nsf0605/nsf0605.jsp
TILE and pio	http://www.mcs.anl.gov/pio-benchmark/ and http://www.mcs.anl.gov/~rross/pio-benchmark/
IOR	http://www.llnl.gov/asci/purple/benchmarks/limited/ior/ior.mpiio.readme.html
HPCS SSCA	http://www.highproductivity.org/SSCABmks.htm
Flash I/O	http://flash.uchicago.edu/~zingale/flash_benchmark_io/

Questions & Lessons learned

- The one number at the end: less important
  → more important: Detailed insight
- Benchmarking the memory cache: one should know what is done
- Non-wellformed / wellformed
- Sequential / random / other patterns
- Overlapping I/O and calculations
- Micro benchmarks versus application-based benchmarks
- Single CPU / parallel with some CPUs / all CPUs
  → Big differences for:
  – Meta data server
  – Disk striping
- Compare benchmarked I/O with physical I/O (or CPU [Linpack]) limits
- Which CPU subset is already able to achieve full physical I/O bandwidth
  → several application may use I/O interleaved
- MPI_File_sync need not write data on physical disk
- **One benchmark can never analyze the total I/O characteristics of a system**

Rolf Rabenseifner

Dr. Rolf Rabenseifner studied mathematics and physics at the University of Stuttgart. Since 1984, he has worked at the High-Performance Computing-Center Stuttgart (HLRS). He led the projects DFN-RPC, a remote procedure call tool, and MPI-GLUE, the first metacomputing MPI combining different vendor's MPIs without loosing the full MPI interface. In his dissertation, he developed a controlled logical clock as global time for trace-based profiling of parallel and distributed applications. Since 1996, he has been a member of the MPI-2 Forum. From January to April 1999, he was an invited researcher at the Center for High-Performance Computing at Dresden University of Technology.

Currently, he is head of Parallel Computing - Training and Application Services at HLRS. He is involved in MPI profiling and benchmarking, e.g., in the HPC Challenge Benchmark Suite. In recent projects, he studied parallel I/O, parallel programming models for clusters of SMP nodes, and optimization of MPI collective routines. In workshops and summer schools, he teaches parallel programming models in many universities and labs in Germany.