Using Non-blocking I/O Operations in HPC to Reduce Execution Times

David Buettner, Julian Kunkel, Thomas Ludwig

Euro PVM/MPI
September 8th, 2009

1 Motivation

2 Theory of a non-blocking I/O benchmark

3 The benchmark and results

4 Future work

Performance improvements

In the computation environment:

- Hardware setup
- MPI distribution
- Parallel file system
- Different tuning options inside the software layers

Inside the application:

- Workload distribution
- Organization of individual program steps
- Non-blocking MPI operations

Performance analysis

In order to improve performance, one needs to

- understand a complex system,
- deal with a lot of unknowns, and
- evaluate a possible margin of profit.

$\Rightarrow$ Benchmarks are needed.

1 Motivation

2 Theory of a non-blocking I/O benchmark

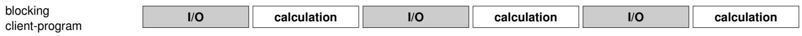
3 The benchmark and results

4 Future work

Possible scenarios

Consider a program which ...

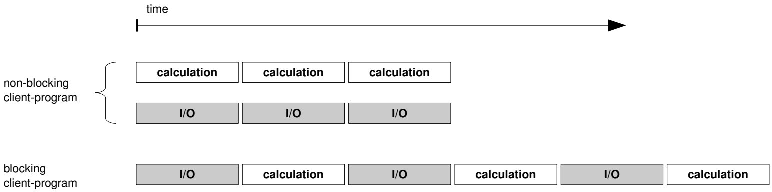
- produces data iteratively,
- does not perform write operations on the data for a while and
- has access to non-blocking I/O functions.



$\Rightarrow$ Hide I/O or calculation behind the other part respectively. Sequential operations become concurrent operations.

Scenarios

The most promising scenario is when I/O and calculation can be overlapped perfectly:



Any other iterative scenario can only save less time.

Evaluation value

`Non-blocking efficiency ratio:`

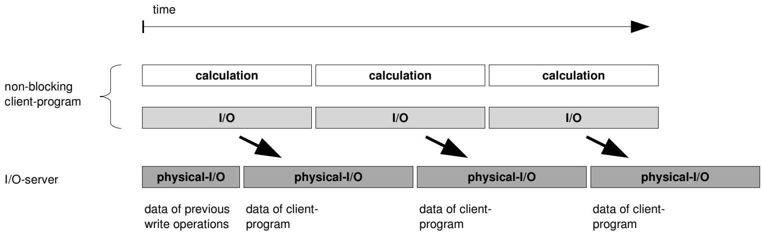
$$0.5 \leq \frac{\texttt{duration of non-blocking I/O version}}{\texttt{duration of blocking I/O version}} (\leq 1.0) \quad (1)$$

The actual value will quantify the overhead necessary to execute I/O and calculation concurrently.

Desired scenario

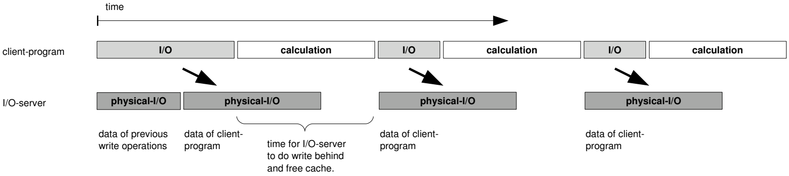
We want a scenario which

- in theory promises best results, and
- requires the file system to do physical I/O as much as possible.



Expected blocking I/O version

- Time on the I/O-server side to free I/O cache
- Shorter execution time of calculation and I/O parts individually



Benchmark goal

Overall, the interest is in

- providing best insight into the actual possible advantages of non-blocking I/O and
- showing how close the used software can get to the optimal efficiency ratio for an optimal scenario.

1 Motivation

2 Theory of a non-blocking I/O benchmark

3 The benchmark and results

4 Future work

Benchmark details

Benchmark sequence:

1. Compute necessary workload so that in the non-blocking version I/O and calculation take the same amount of time
2. Run non-blocking test
3. Run blocking test

Before each phase fill file system buffer.

Computing environment

Hardware:

9 nodes with

- Two Intel Xeon 2GHz CPUs
- 1GB DDR-RAM
- 1-GBit/s-Ethernet-Interfaces

5 I/O nodes used for hosting the pvfs2-servers, each with:

- two 160 GB S-ATA HDDs set up as a RAID-0.

Computing environment

Software:

- MPICH2 in version: mpich2-1.0.5p4
  non-blocking functions emulated using threads
- PVFS2 in version 2.6.2

For visualization of the benchmark behavior we used PIOviz, which includes a modified MPE trace and extended Jumpshot.

Computing environment

Software:

- MPICH2 in version: mpich2-1.0.5p4
  non-blocking functions emulated using threads
- PVFS2 in version 2.6.2

For visualization of the benchmark behavior we used PIOviz, which includes a modified MPE trace and extended Jumpshot.

Test combinations

Executed tests include the combinations of:

- Number CPUs per benchmark process: 1, 2
- Number benchmark processes: 1,2,4
- Number I/O-servers: 1,2,4

Test combinations...

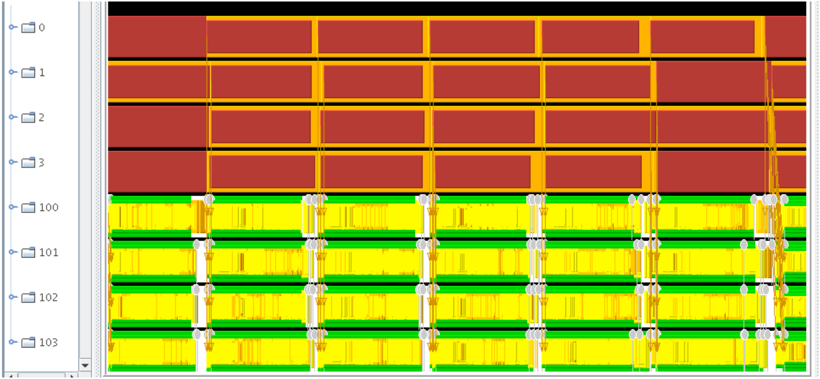
2 CPUs per benchmark process in order to ignore possible overhead.

These test showed that theoretical optimum can nearly be achieved:

$$\texttt{non-blocking efficiency ratio} = 0.53 \qquad (2)$$

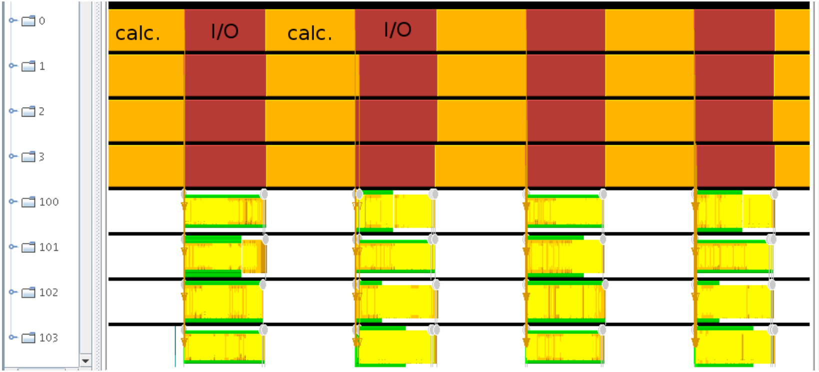
Non-blocking I/O test

4 CPUs, 4 benchmark processes, 4 I/O-servers



Blocking I/O test

4 CPUs, 4 benchmark processes, 4 I/O-servers



Scenario results

In average over a series of test runs we have:

$$\texttt{non-blocking efficiency ratio} = 0.67 \tag{3}$$

In the non-blocking I/O test,

- the calculation time increases by a factor of 1.25 and
- the I/O time increases by a factor of 1.29

compared to the times needed for each in the blocking I/O test.

Overall results...

For all interesting cases with 1CPU per benchmark process we obtained:

$$\texttt{non-blocking efficiency ratio} < 0.75 \tag{4}$$

Allowing for the individual parts to take longer in the non-blocking version.

Conclusion

While we only looked at:

- write operations and
- scenarios well suited for the use of non-blocking I/O,

non-blocking I/O operations show potential to reduce execution times of HPC applications.

- Performance improvements with non-blocking I/O is possible with the used computing environment.
- Measurements suggest that short I/O phases can be hidden entirely behind longer calculation phases.

Future work

Interesting will also be an analysis of

- read operations,
- large scale tests,
- scenarios where either calculation or I/O take up the bulk of the execution time.