PCJ-BLAST – massively parallel sequence alignment using NCBI Blast and PCJ Java library

Piotr Bała bala@icm.edu.pl
ICM University of Warsaw

Marek Nowicki faramir@mat.umk.pl
ICM University of Warsaw
N. Copernicus University

Davit Bzhalava davit.bzhalava@ki.se
Karolinska Instituet

Blast

- Sequence alignment is essential for NGS
- There is a number of software packages for sequence alignment based on various a similarity search methods
- BLAST Basic Local Alignment Search Tool (1991)
  - The heruistic algorithm it uses is much faster than other approaches
  - The search time can be long (days or weeks) for large datasets
- NCBI blast is the most widely used implementation

- There is strong interest in using large computer systems to run blast
  - Blast runing on cloud or grid
  - Parallel versions of blast running on HPC systems

Fasta input (2 reads out of 1 milion)

>C1093377_2.0

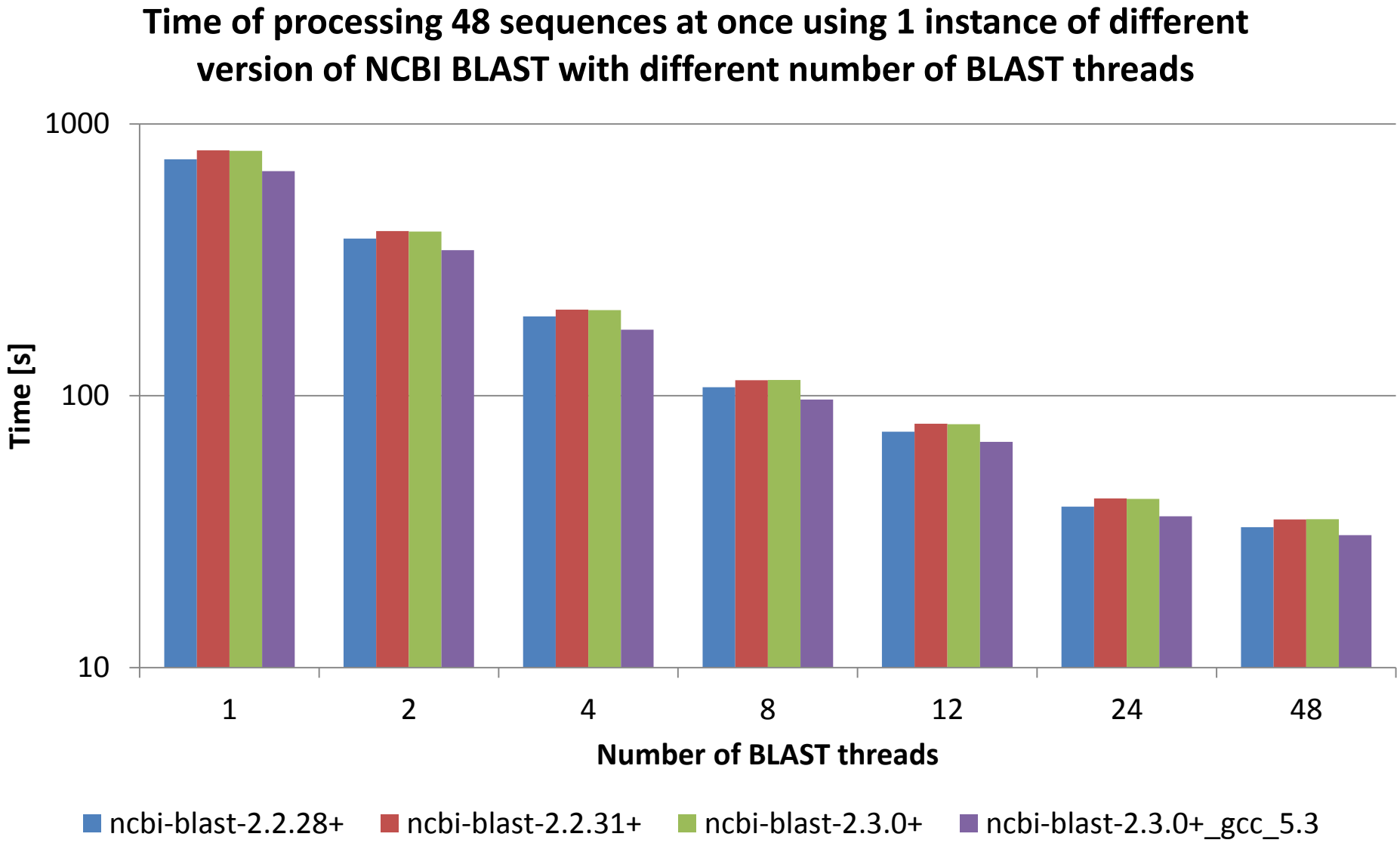
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAACACTTTGGGAGGCCAAGATGGGAAGATCTTTTGAGG
CCAGGCGTTCAAGACCAGTCTGGGCAATATGGAGAGACCT
GTCTCTACAAAAAAAATTAGCCAGACCTAGTGGCTGGCTGA
GGCAGGAGGATCATCTGAGCTCAGGAGATTGAGAT

>C1093379_2.0

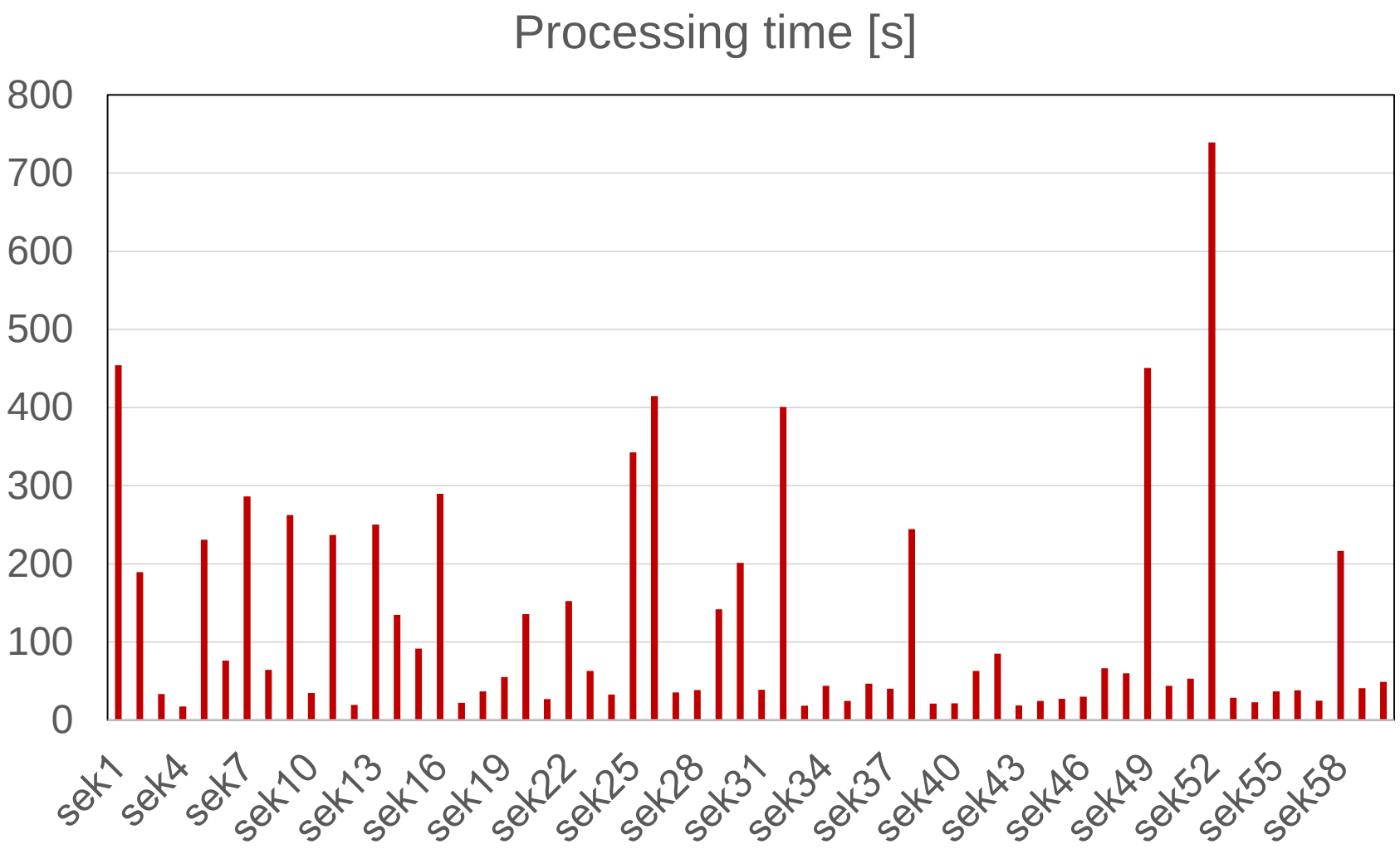
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAATTTTAAGTTTTTCCTTATAAGGGAAATAAGCTTATTTCT
TTTAGAAGCAGAAACTGCAAATGGGGGTGTATGGAATTCTTT
TTTTTTTTTTTTTTTTTTGCGACAAGGTCTCACTGTGTTGCCCA
GACTGGAGTGCAGTGGTGCAATCATAGCT

…

NCBI blast performance (XC40 single node)



Blast – processing time of reads changes



Parallel blast implementations

- mpiBLAST – uses old version (2012) of blast
- parallel_blast.py – limited to single node, regular split of input
- Paracel BLAST (based on NCBI blast 2.2.26) – costly!
- The BlueGene implementation – limited to IBM BGQ systems
- pioBLAST parallel version which uses MPIIO – memory problems while reference database is large

- SparkBLAST – Apache Spark implementation running on AWS cloud (PhD thesis 2016)

Pcj-blast

- Parallel version developed using PCJ (Parallel Computing in Java library developed at ICM)
- Java code is responsible for reading fasta input and execute NCBI-blast on set of sequences (reads)
- The NCBI blast (currently 2.2.28) is executed with the `-num_threads` option
  - For 28 core nodes: 4 PCJ threads with –num_threads=7

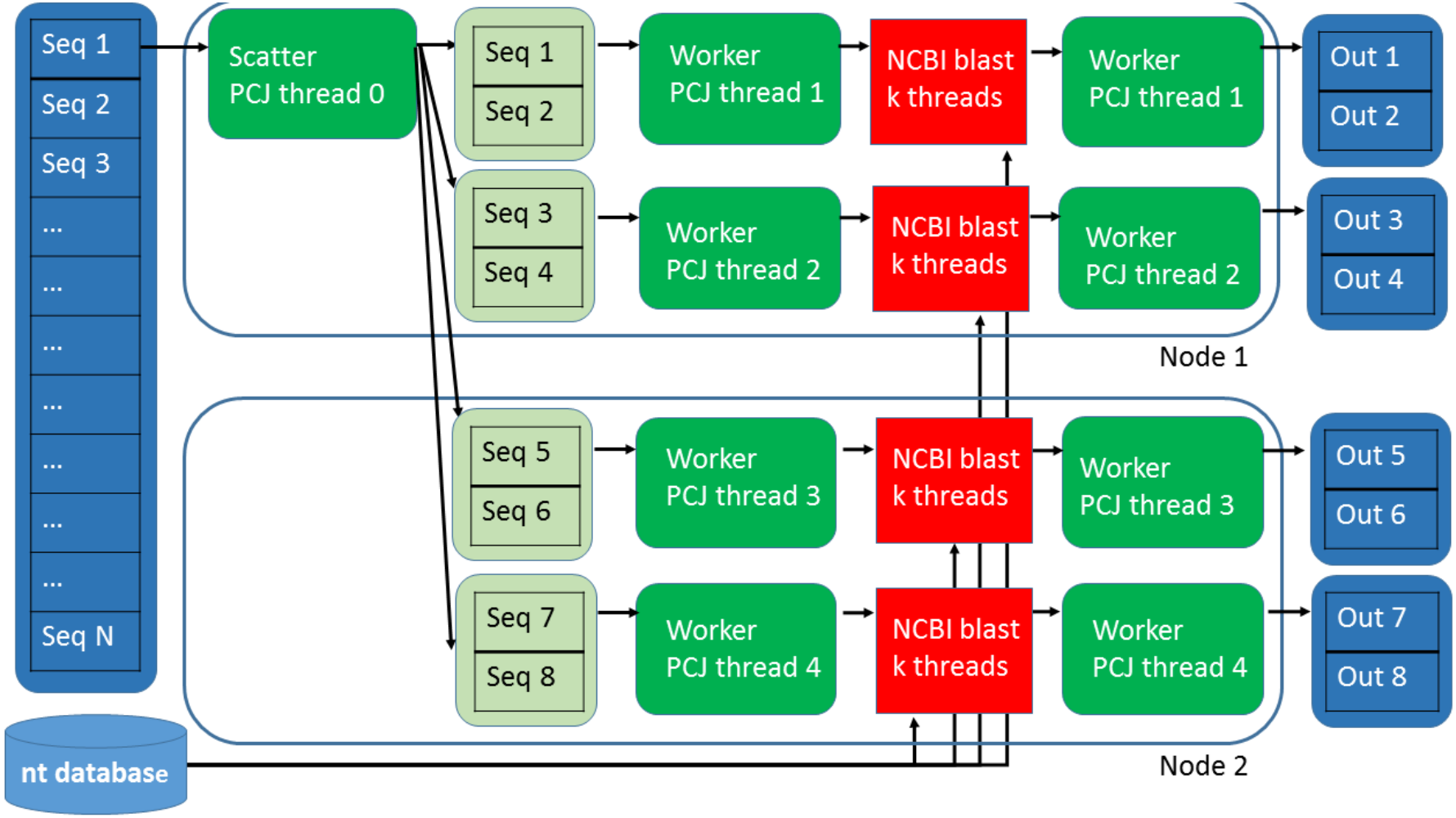
- The solution can be executed on any multinode system with Java 8 installed
- The I/O performance is important
  - thousands of blast instances reads blast database (52GB) concurrently
  - lustra filesystem performs worse than nfs

Pcj-blast

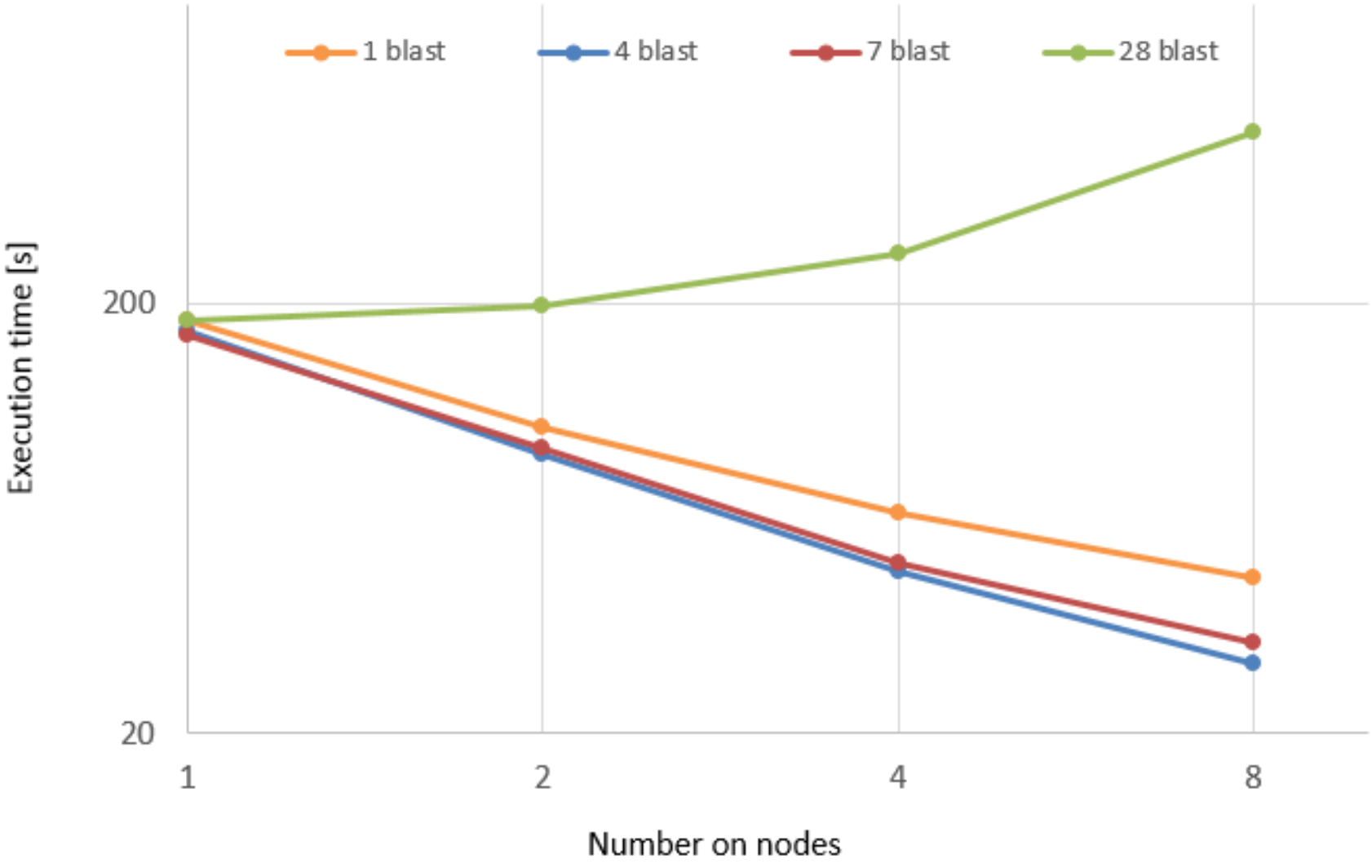
- Reads input sequence (fasta) line by line
- Blocks of lines (reads) are used as input for blast query
- Pcj-blast can start hundreds of blast instances
- NCBI blast is used
  pcj-blast not bounded to particular version
- One or more NCBI blast instances are run on the hardware node
- Output from each node (text or XML) is gathered
- Output is postprocessed and is stored in a single file
  - Postprocessing is parallelized

- Pcj-blast can be run on HPC systems, clusters
  - Requires Java 8 and NCBI blast installed

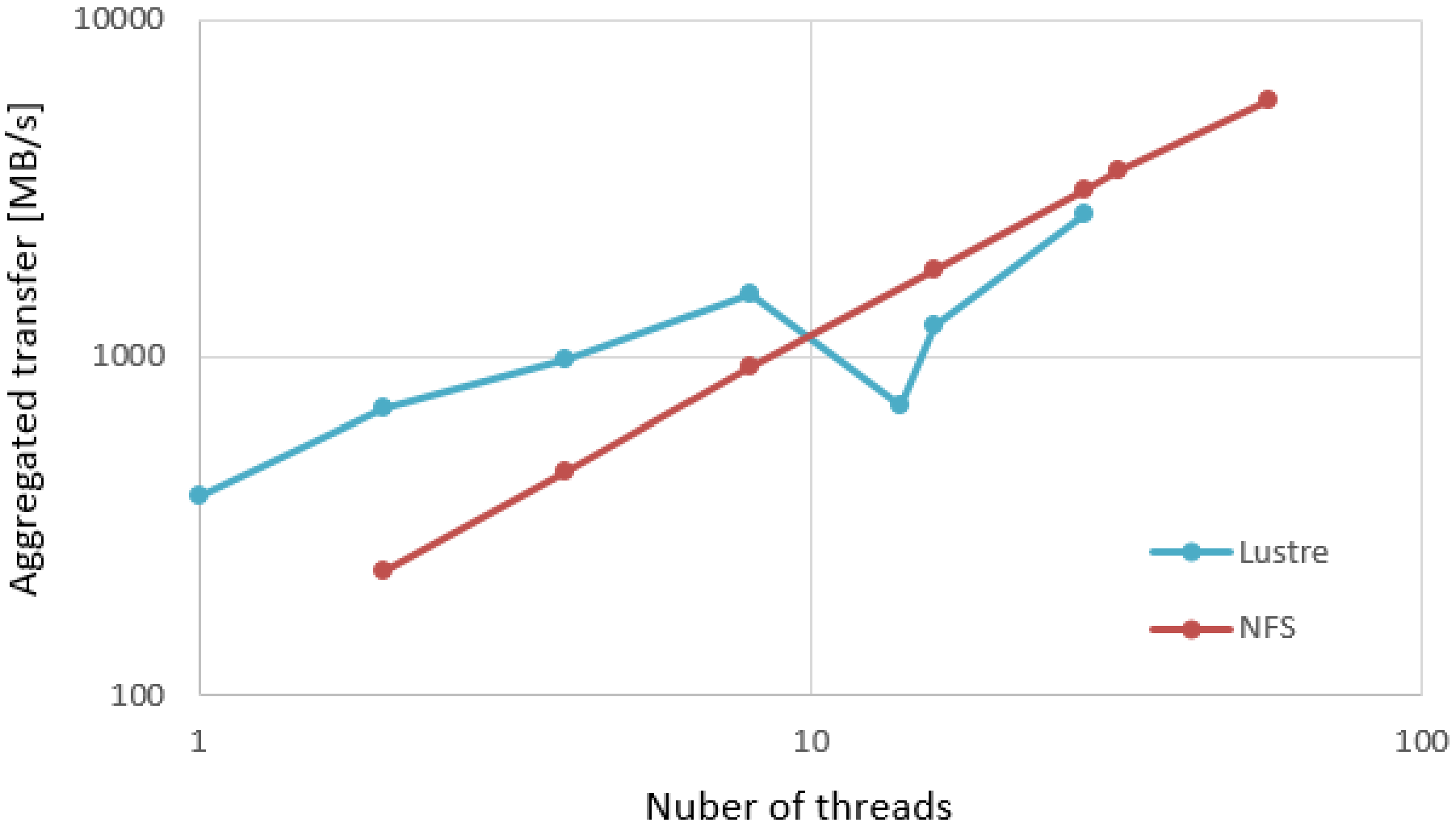
pcj-blast


Seq 1
Seq 2
Seq 3
...
...
...
...
...
...
...
Seq N
nt database
Scatter
PCJ thread 0
Seq 1
Seq 2
Seq 3
Seq 4
Seq 5
Seq 6
Seq 7
Seq 8
Worker
PCJ thread 1
Worker
PCJ thread 2
Worker
PCJ thread 3
Worker
PCJ thread 4
NCBI blast
k threads
NCBI blast
k threads
NCBI blast
k threads
NCBI blast
k threads
Worker
PCJ thread 1
Worker
PCJ thread 2
Worker
PCJ thread 3
Worker
PCJ thread 4
Out 1
Out 2
Out 3
Out 4
Out 5
Out 6
Out 7
Out 8
Node 1
Node 2

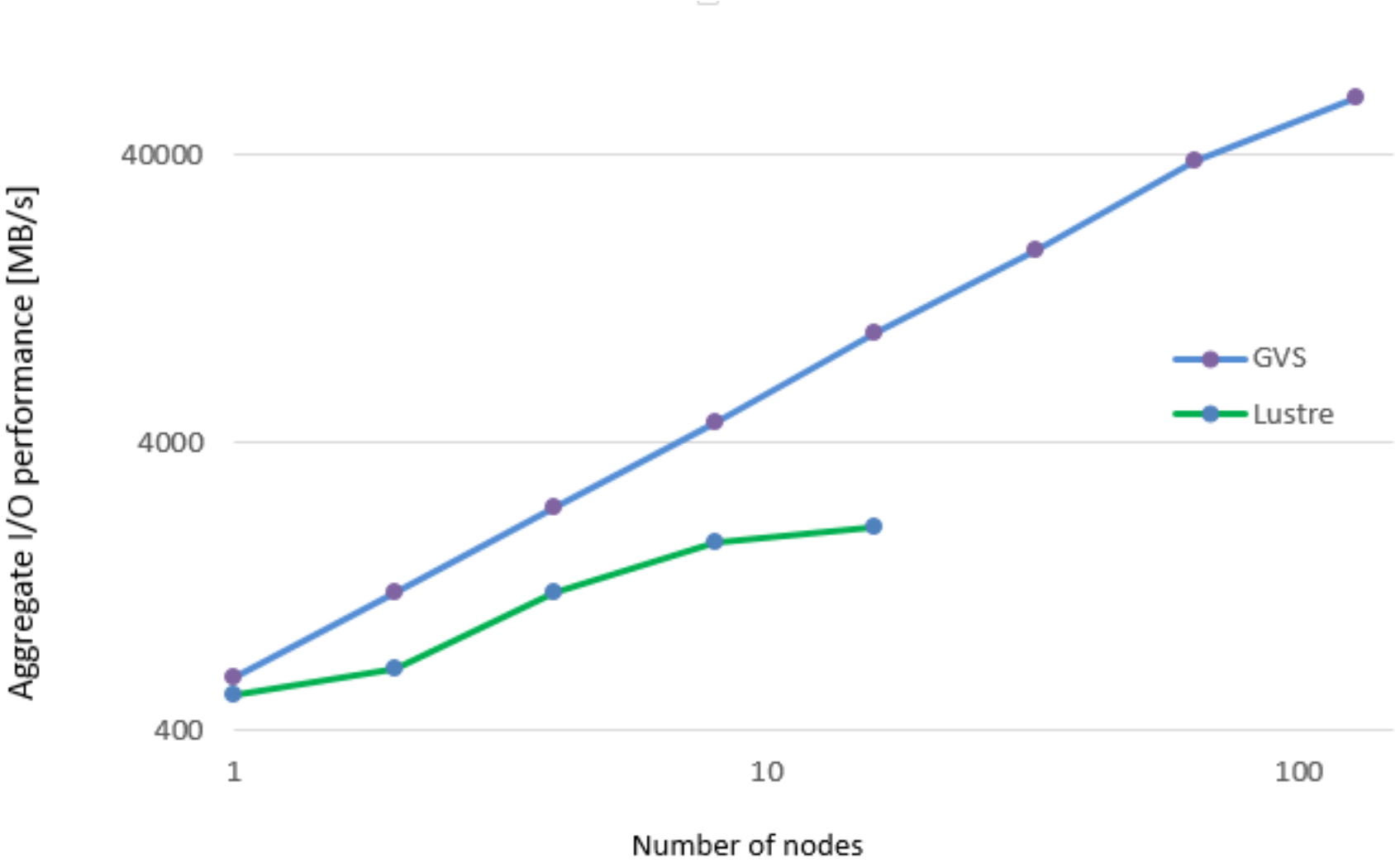
Multiple NCBI blast running on single node (X86 cluster)



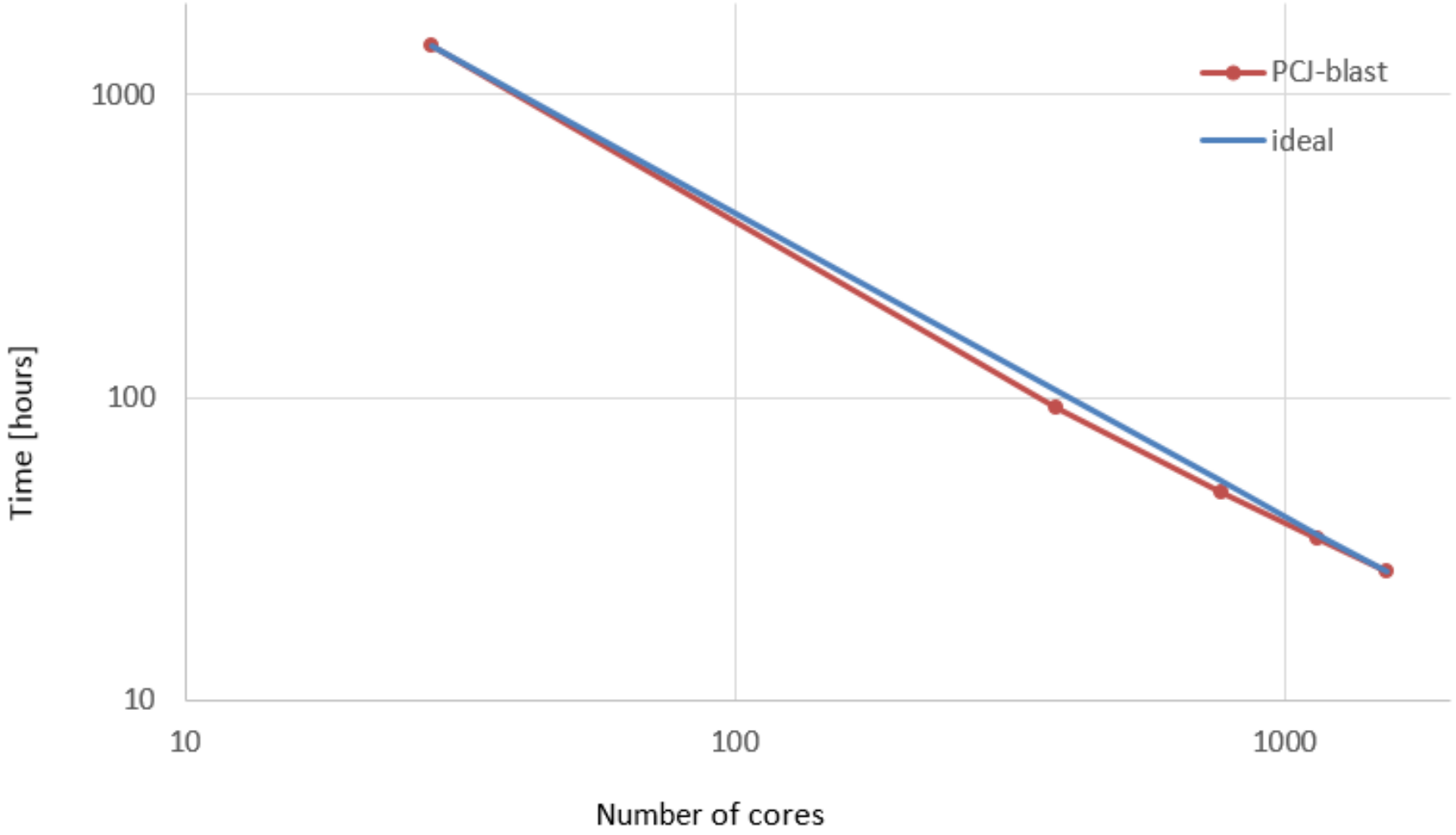
I/O performance (x86 single file read)



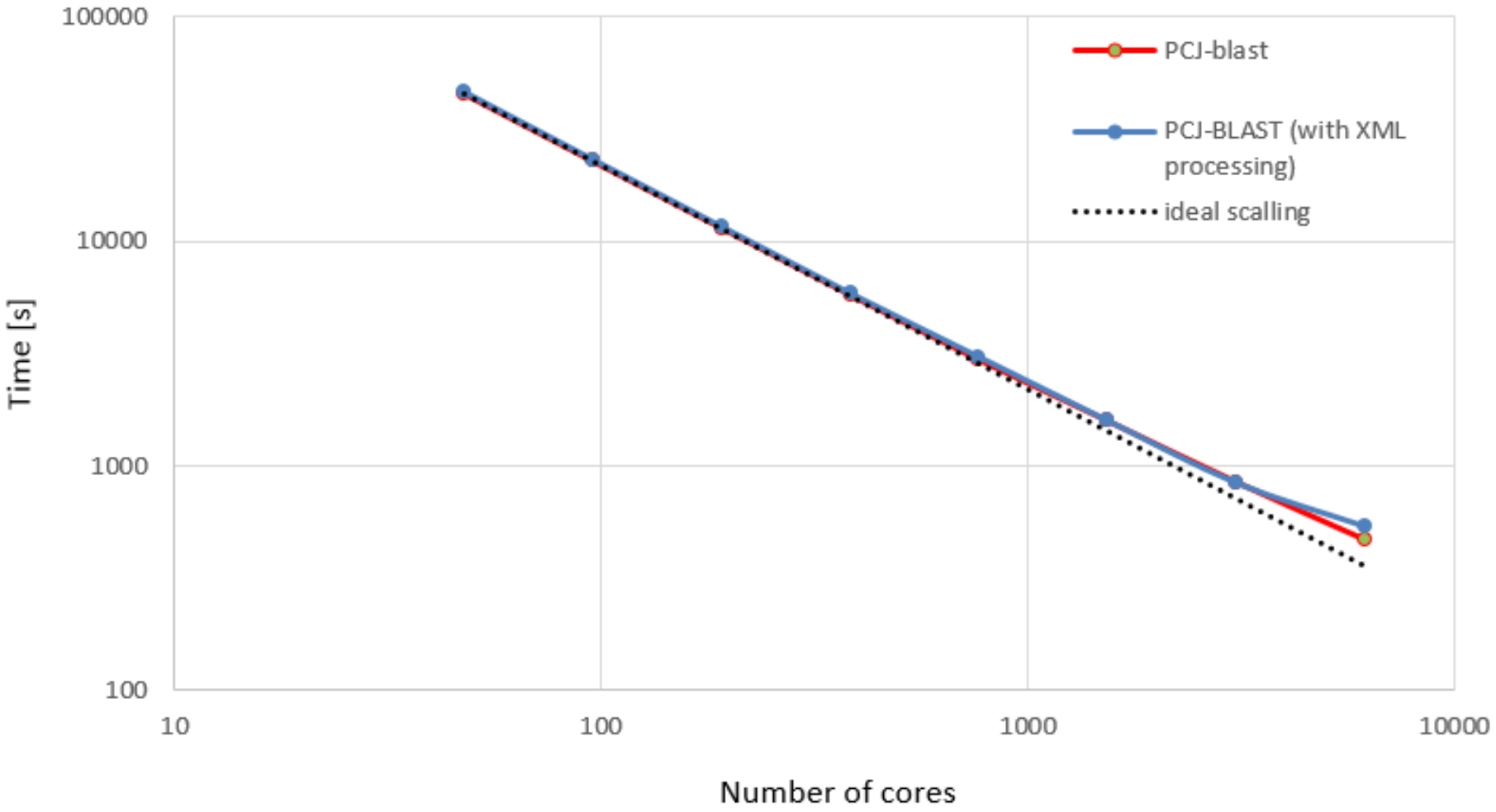
I/O performance (XC40 single file read)



pcj-blast scalability (x86 cluster)



pcj-blast scalability (Cray XC40)



PGAS languages

- http://www.pgas.org

- Independent threads
- Local and shared (global) variables
- Global variables visible to all threads
- One-sided communication
  - Communication details hidden to programmer
- Main operations:
  - Synchronization (barrier)
  - get
  - put

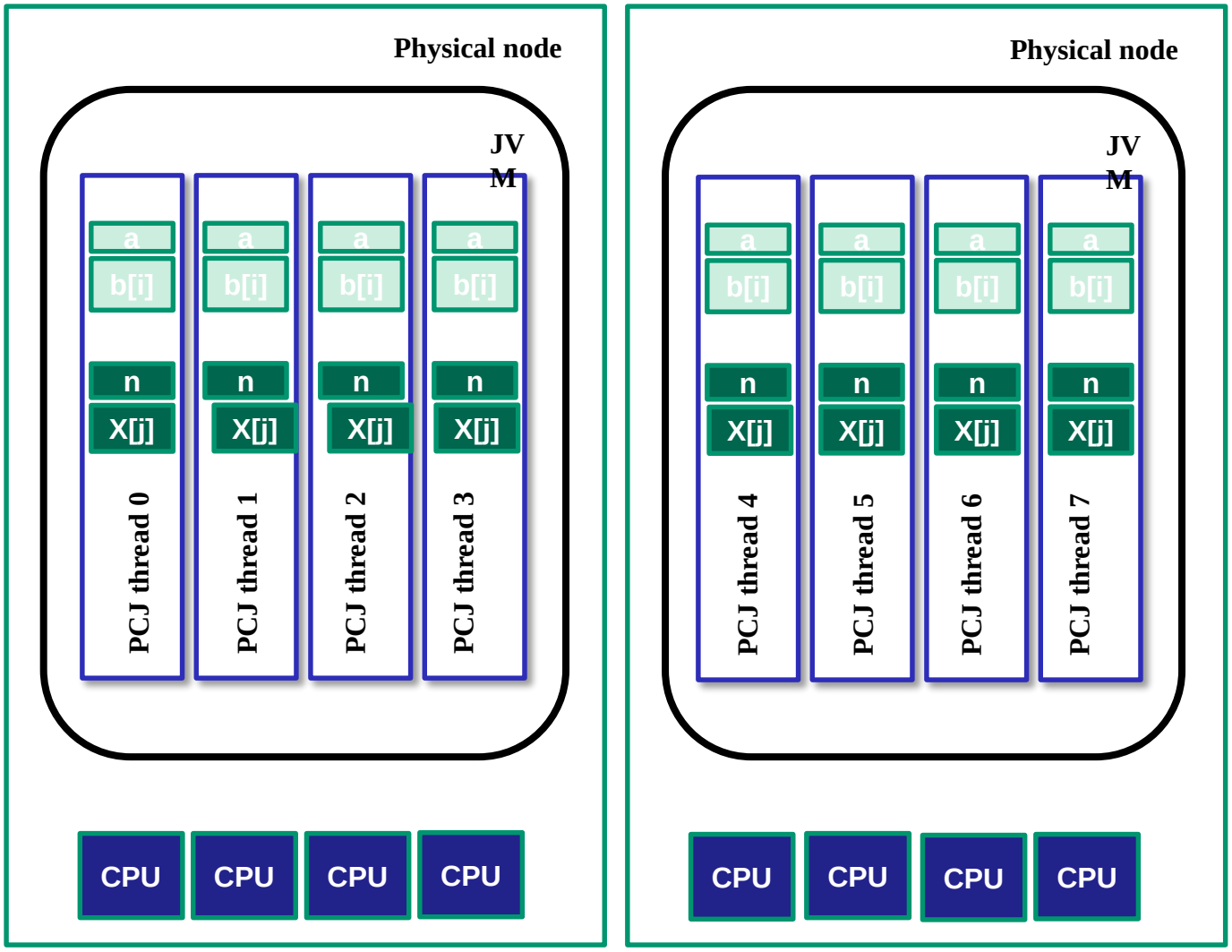
PCJ Library

Java library

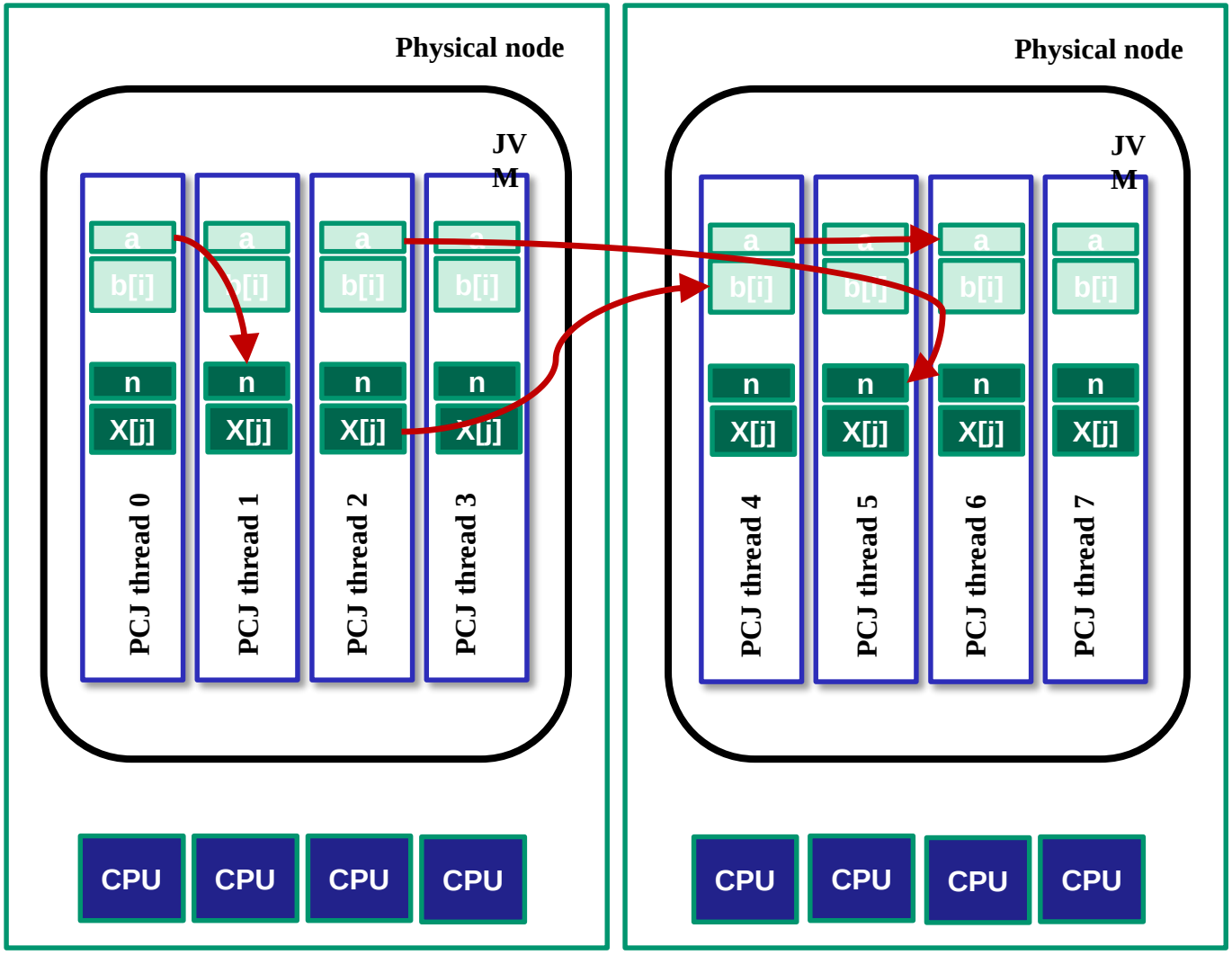
- **pcj.icm.edu.pl**

- Designed based on PGAS (Partitioned Global Address Space) paradigm
- Simple and easy to use
- Does not introduce extensions to the Java language
  - no new compilers nor pre-processor
- Does not require additional libraries
- Works with Java 1.8, 1.9
  - Version for Java 1.7 available
- Good performance
- Good scalability (already beyond 50k+ cores)

PCJ – memory layout



PCJ – memory layout and communication



PCJ – Parallel Computations in Java

Main functionality:

- Synchronization
- Get data from other thread (get)
- Send data to other thread (put)

Additional functionality:

- Broadcast
- Monitoring of the variable change (useful with put())
- Parallel I/O
- Groups of threads

PCJ – Hello world

```
import org.pcj.*
public class PcjHelloWorld implements StartPoint {

   @Override
   public void main() {
        System.out.println("Hello!");
   }

   public static void main(String[] args) {
      PCJ.deploy(PcjHelloWorld.class,
                        new NodesDescription("nodes.txt"));
   }
}
```

PCJ – main methods

```
@Shared(ExampleGetPut.class)     // Define storage
   enum Shared { a, array }

if (PCJ.myId()==0) c =(double) PCJ.get(3, Shared.a);

if (PCJ.myId()==0) PCJ.put(5.0, 3, Shared.a);
if (PCJ.myId()==0) PCJ.put(5.0, 3, Shared.array, p);

if (PCJ.myId()==0) PCJ.broadcast(5.0, Shared.a);

public static void PCJ.barrier();
public static int PCJ.threadCount()
public static int PCJ.myId()
```

PCJ – read data (seqencial)

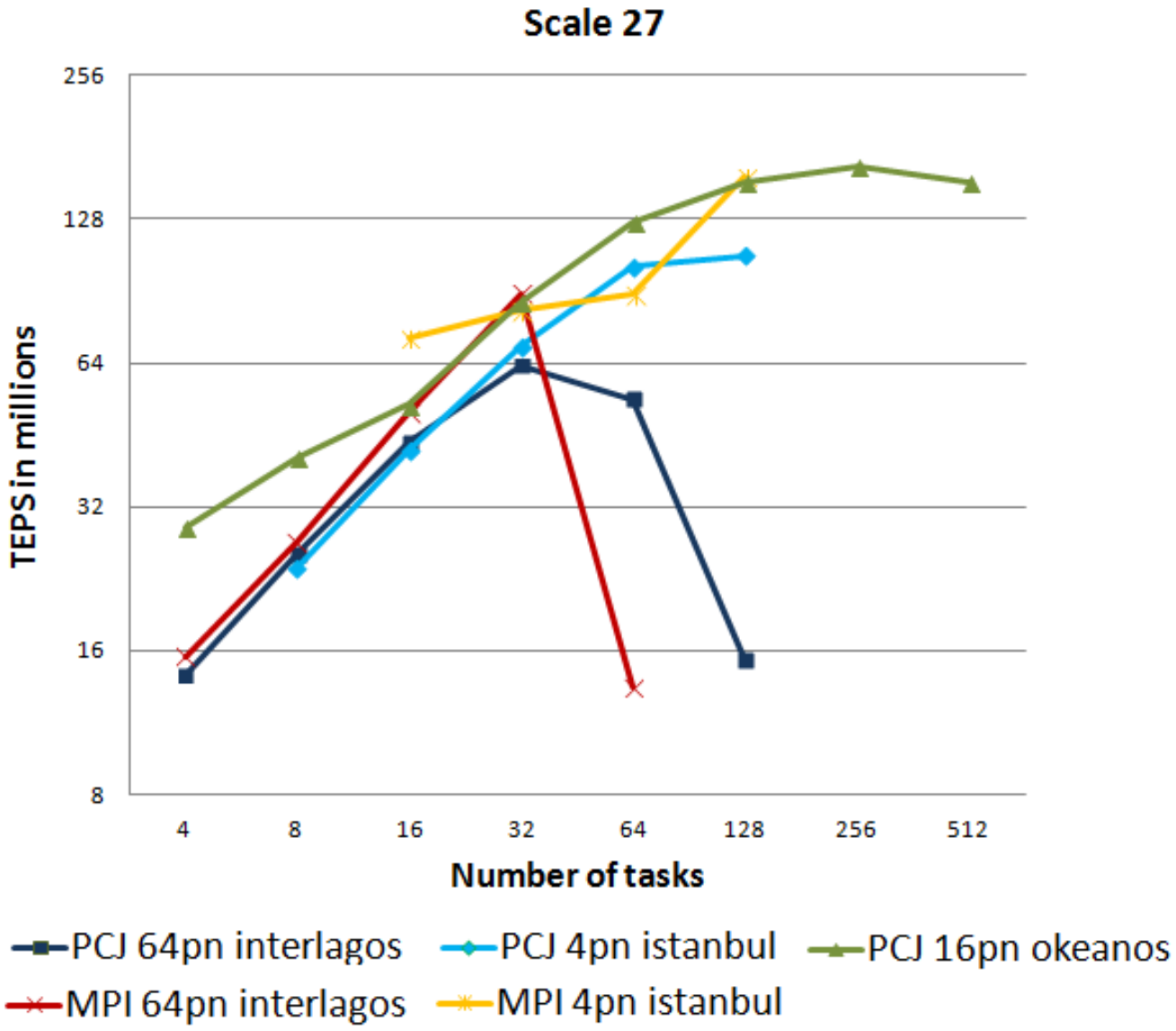
```
@Shared(ExampleRead.class)     // Define storage
    enum Shared { a }
    double a;

 double al;
if (PCJ.myId()==0) {
          Scanner s = new Scanner(System.in);
          al = s.nextDouble();
          PCJ.broadcast(al., Shared.a);
} else {
         PCJ.waitfor(Shared.a);
}
PCJ.barier()
```

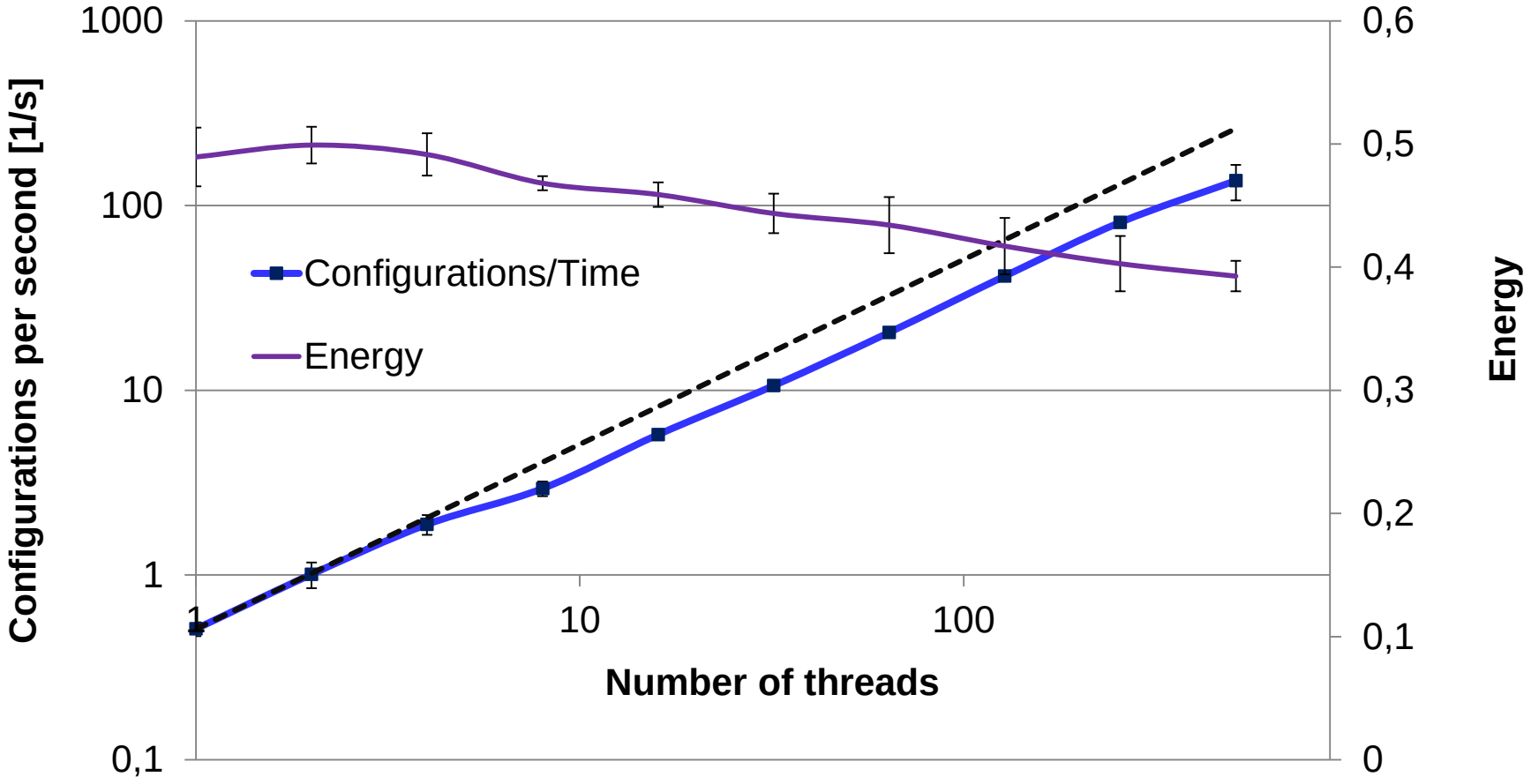
PCJ – Gather using PCJ.asyncGet()

```
@Shared (PcjGather.class)
   enum Shared { a }
   double a
PcjFuture aL[] = new PcjFuture[PCJ.threadCount()];
double sum = 0.0;
 if (PCJ.myId() == 0) {
      for (int p = 0; p < PCJ.threadCount(); p++) {
         aL[p] = PCJ.asyncGet (p, Shared.a);
      }
      for (int p = 0; p < PCJ.threadCount(); p++) {
         sum = sum + (double) aL[p].get();
      }
}
```

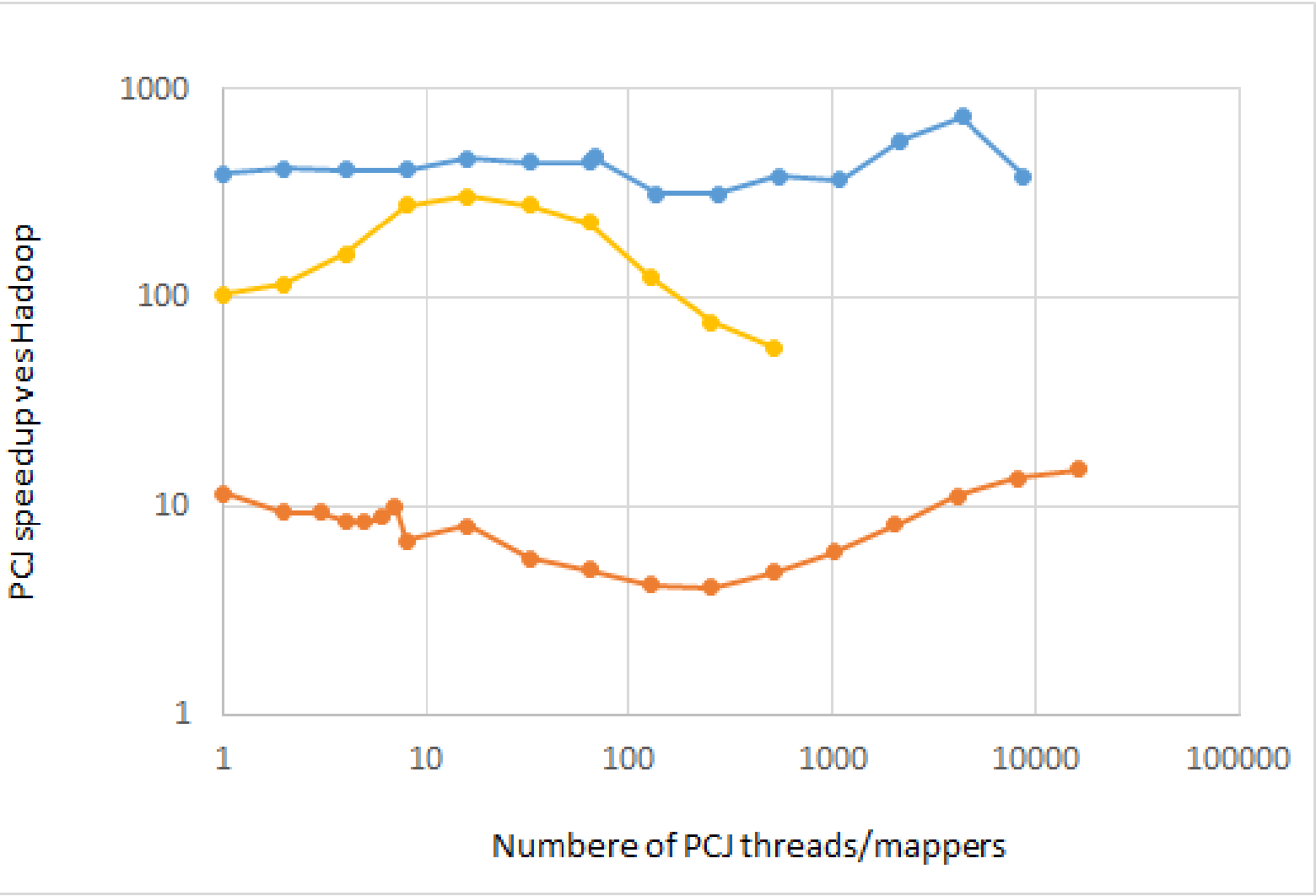
Graph500 – kernel 2 (BFS) Cray XC40



Neural network optimization using GA (conectom of *C. Elagans)*



PCJ vs Apache Hadoop (newton @ ICM)



Pi Estimate (blue), BFS (orange), WordCount (red)

PCJ-BLAST – massively parallel sequence alignment using NCBI Blast and PCJ Java library

PCJ: pcj.icm.edu.pl

Piotr Bała bala@icm.edu.pl
ICM University of Warsaw

Marek Nowicki faramir@mat.umk.pl
ICM University of Warsaw
N. Copernicus University

Davit Bzhalava davit.bzhalava@ki.se
Karolinska Instituet