Massively Parallel Sequence Alignment using pcj-blast

dr Marek Nowicki[2], prof. Piotr Bała[1]
bala@icm.edu.pl

[1] ICM University of Warsaw, Warsaw, Poland
www.icm.edu.pl

[2] Nicolaus Copernicus University in Toruń, Poland

Motivation

- Try to find similarities with sequences database
  - 52 GB
  - 675 million lines of nucleotide sequences (over 20 millions sequences)

- Sample FASTA input (3 sequences/reads - of out 1 million)

```
>C1093377_2.0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACACTTTGGGAGGCCAAGATGGGAAGA
TCTTTTGAGGCCAGGCGTTCAAGACCAGTCTGGGCAATATGGAGAGACCTGTCTCTACAAAAAAATTAGCCAG
ACCTAGTGGCTGGCTGAGGCAGGAGGATCATCTGAGCTCAGGAGATTGAGAT
>C1093379_2.0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTTTAAGTTTTTCCTTATAAGGGAAAT
AAGCTTATTTCTTTTAGAAGCAGAAACTGCAAATGGGGGTGTATGGAATTCTTTTTTTTTTTTTTTTTTTGCG
ACAAGGTCTCACTGTGTTGCCCAGACTGGAGTGCAGTGGTGCAATCATAGCT
>C1093381_1.0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGATATTTCTCATTATGGTCATACTTTA
ACTTTCTATAGTTCAGTATTTAGTCTCCTGTTTCAATTGTGCCAGATGTAAATTTTCATTTTTTTTTCATTTTG
GATTAGAACCAAAGATATATGTTCAGATAAGGCGAACTTCTATCTATGGAAC
...
```

Motivation

- Sequence alignment is essential for Next Generation Sequencing (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 heuristic 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
- NCBI-BLAST is for mamy researchers reference implementation

Motivation

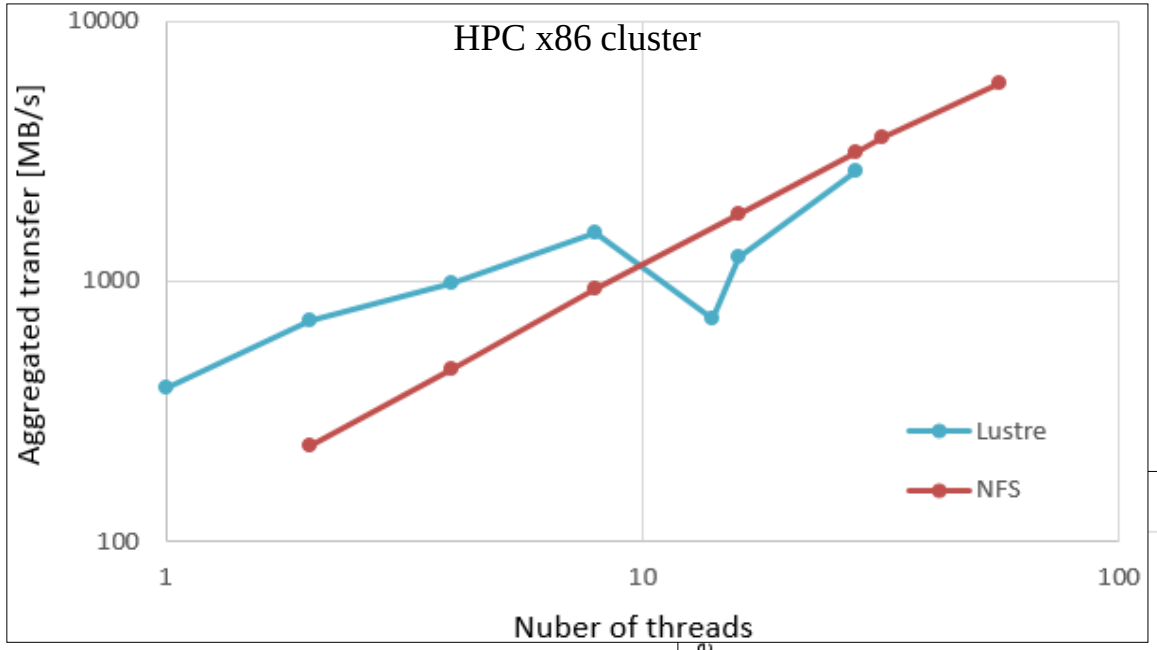
- There is strong interest in using large computer systems to run BLAST
  - mpiBLAST
    - based on old version of BLAST (2012)
  - pioBLAST
    - MPI-IO
    - memory problems while reference database is large
  - Paracel BLAST
    - costly!
  - SparkBLAST
    - Apache Spark implementation running on AWS cloud (PhD thesis 2016)

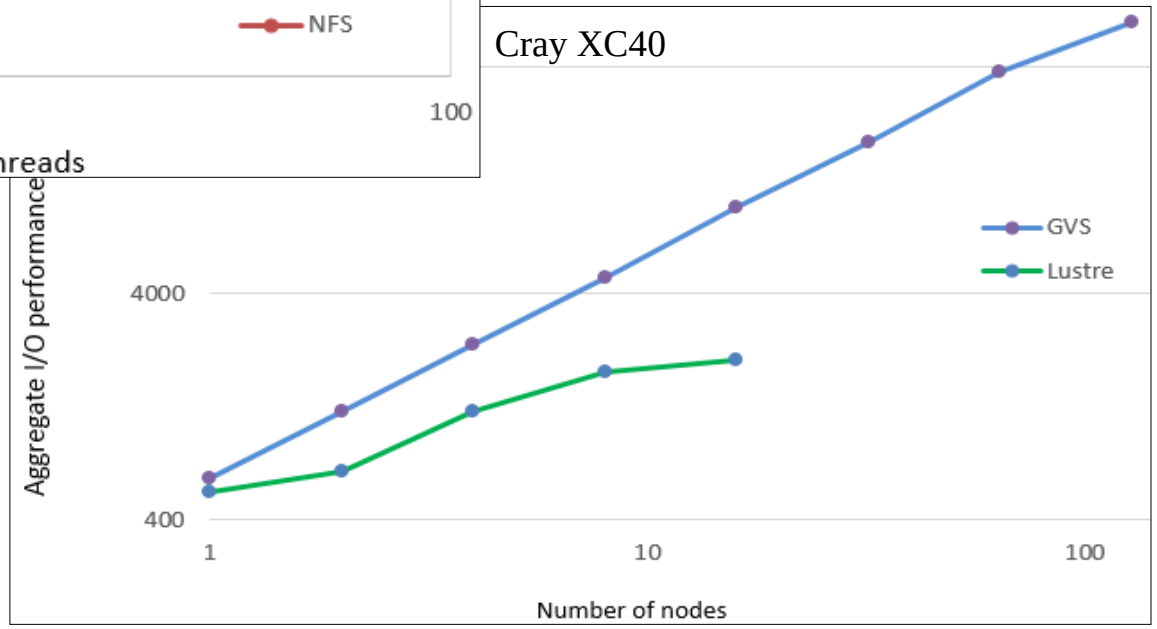
PCJ-BLAST

- Implemented in Java
- Uses PCJ library
- Runs on workstations, clusters, supercomputers
  - can be executed on any system with Java 8
- Reads FASTA input file
- Distributes sets of sequences
- Starts multiple NCBI-BLAST in parallel
  - not bound to particular version
  - one or more instances on the hardware node
- Can concurrently post-process output
  - or just leave it untouched

PCJ-BLAST

- The I/O performance is important
  - thousands of BLAST instances reads database (52GB) concurrently
  - Lustre filesystem performs worse than NFS





PCJ

Java library (open source, BSD license)

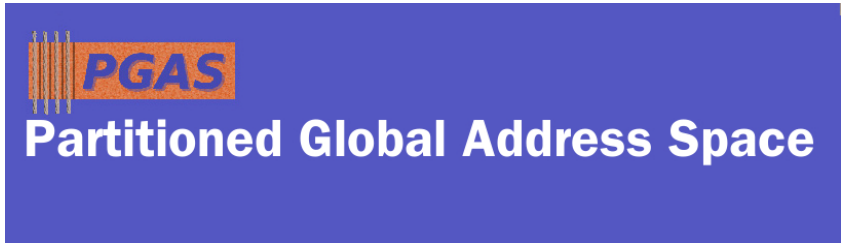
- **`http://pcj.icm.edu.pl`**

PCJ Parallel Computing in Java

- Designed based on PGAS (Partitioned Global Address Space) paradigm
- Small (1 jar file, ~170 KB), simple, and easy to use
- Does not introduce extensions to the Java language
  - no new compilers nor pre-processor
- Does not require additional libraries
  - no problem with dependencies
- Works with Java 1.8 (and up)
  - Version for Java 1.7 available
- Good performance
- Good scalability (already beyond 200k+ cores)

PCJ & PGAS

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

Main functionality:

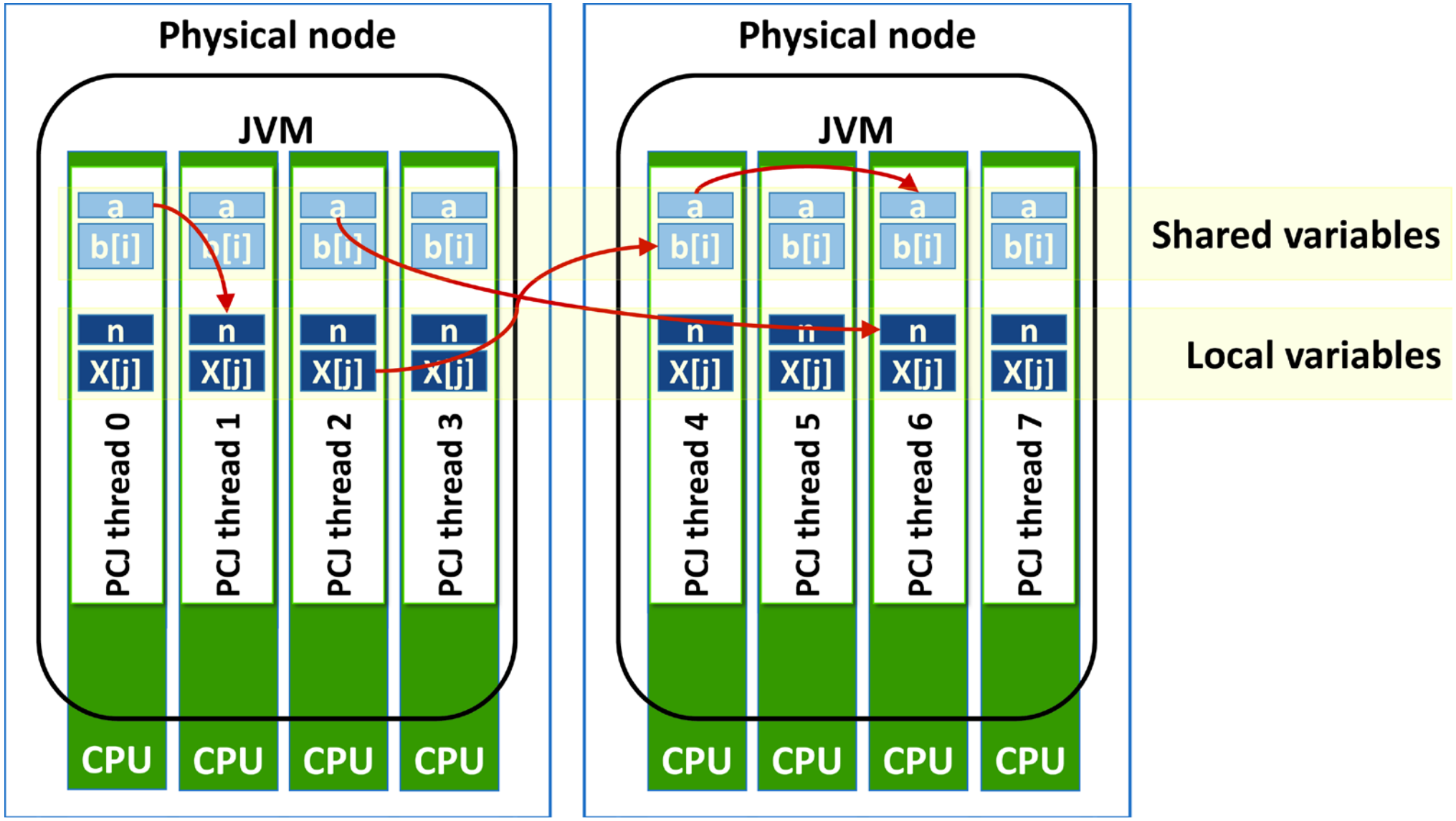
- Synchronization
- Get data from other thread (get)
- Send data to other thread (put)

PCJ Parallel Computing in Java

Additional functionality:

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

PCJ – memory layout and communication

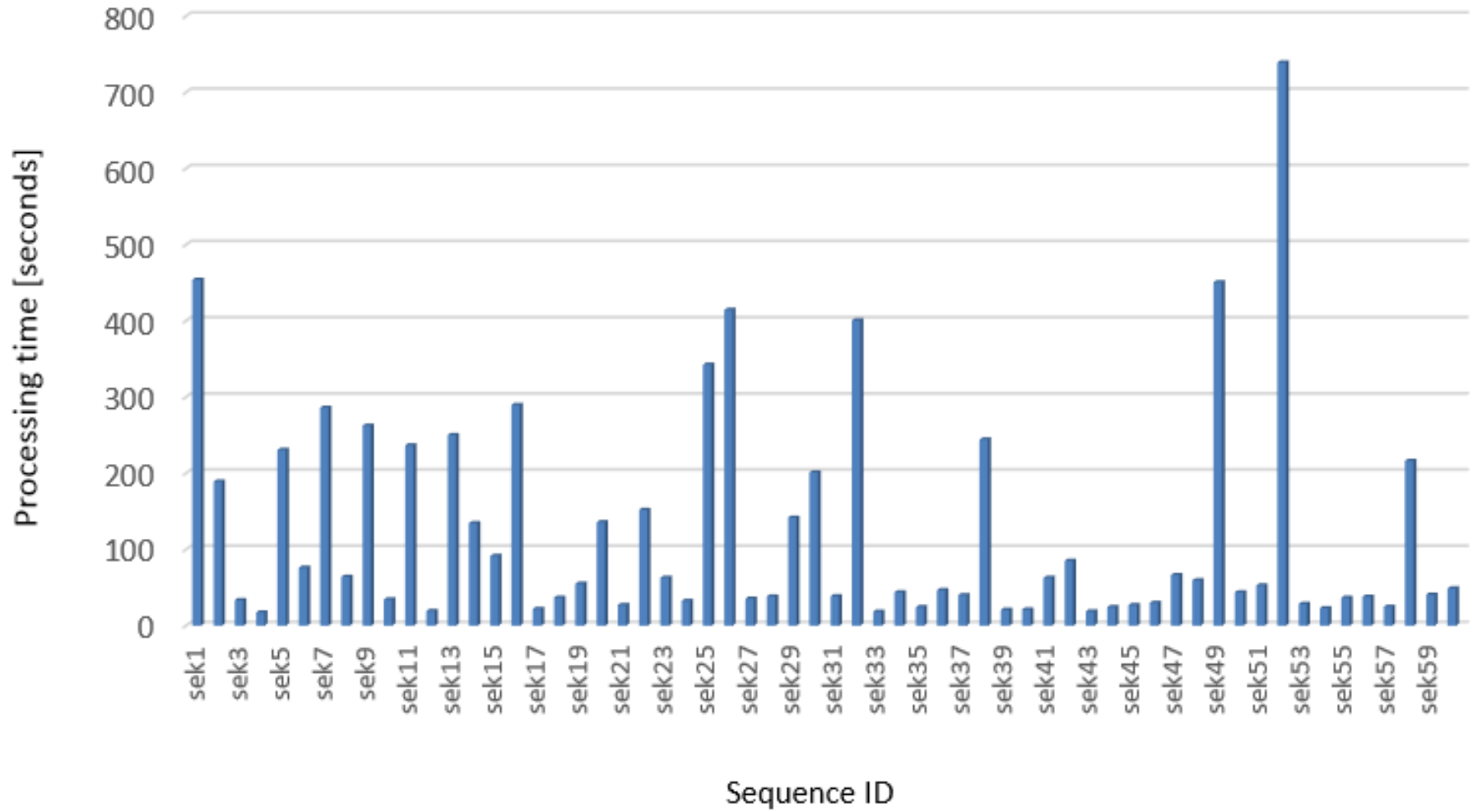


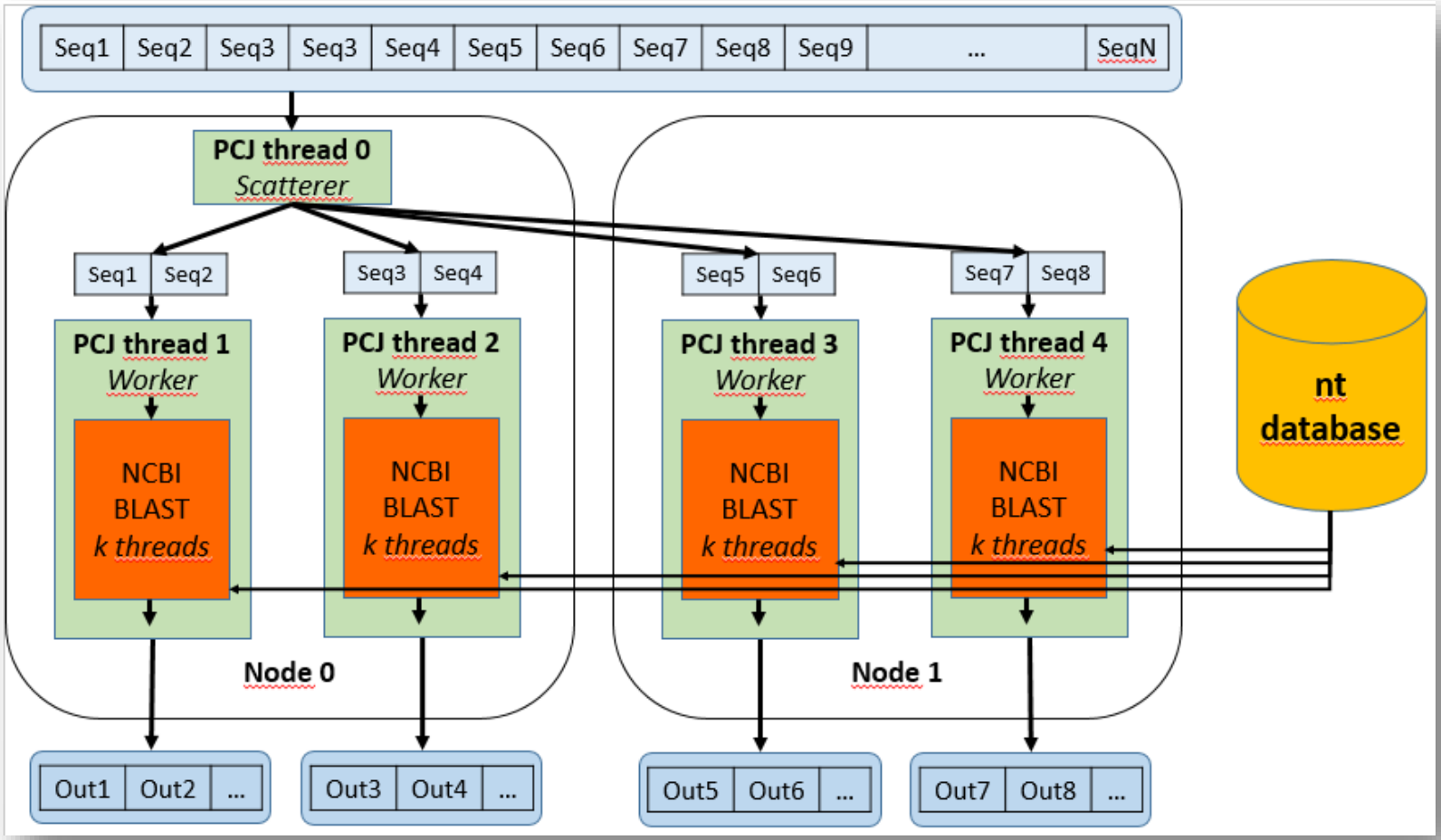
PCJ - Parallel Computing in Java

Package: **`org.pcj`**

- StartPoint ← interface; indicate start class/method
- NodesDescription ← description of nodes
- PCJ ← contains base static methods
  - start / deploy ← starts application
  - myId / threadCount ← get thread identifier / thread count
  - registerStorage ← registers shared space
  - get / asyncGet ← get data from shared space
  - put / asyncPut ← put data to shared space
  - broadcast / asyncBroadcast ← broadcast data to shared space
  - waitFor ← wait for modification of data
  - barrier / asyncBarrier ← execution barrier
  - join ← create/join to subgroup
- PcjFuture<T> ← for notification of async method
- @Storage ← shared space annotation
- @RegisterStorage ← registering shared space

PCJ-BLAST


800
700
600
500
400
300
200
100
0
Processing time [seconds]
sek1
sek3
sek5
sek7
sek9
sek11
sek13
sek15
sek17
sek19
sek21
sek23
sek25
sek27
sek29
sek31
sek33
sek35
sek37
sek39
sek41
sek43
sek45
sek47
sek49
sek51
sek53
sek55
sek57
sek59
Sequence ID



- DNA sequence alignment
- PCJ-Blast – wrapper to NCBI-BLAST
- Performs dynamic load-balancing
- 2x faster than partition of the query and database

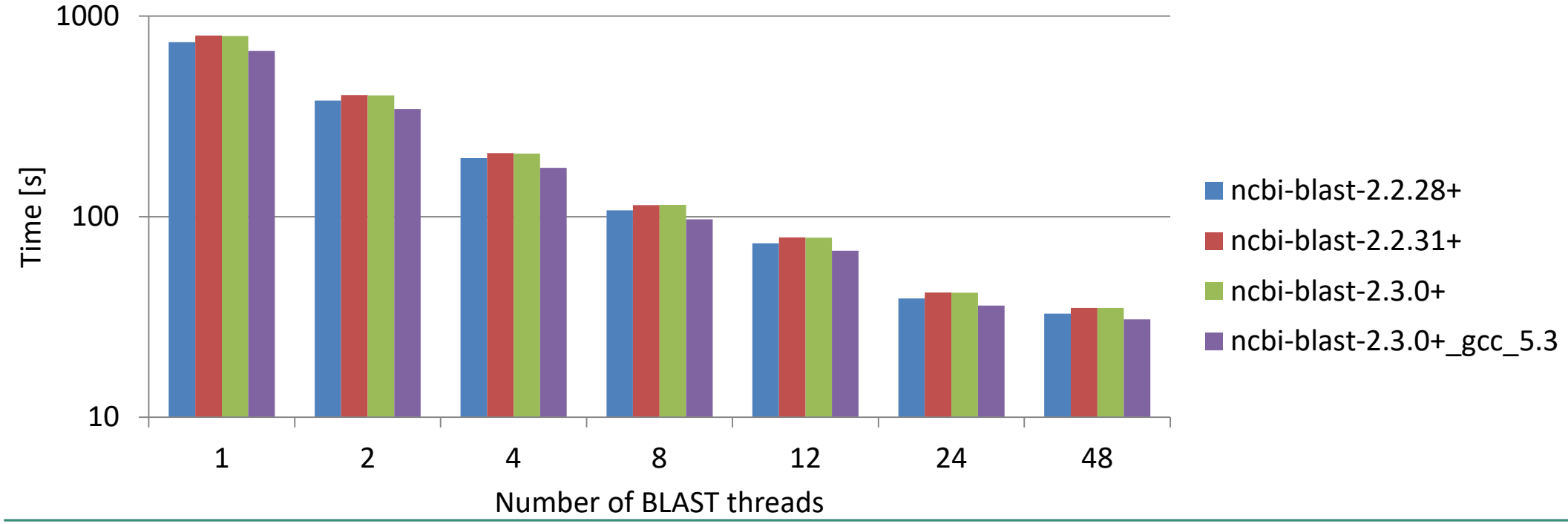
Results

- Used systems:
  - HPC x86 cluster
    - Processor: Intel Xeon E5-2697 v3 CPU @ 2.6 GHz (28 cores)
    - Memory: 64 GB
    - Interconnection: Infiniband FDR, Gigabit Ethernet
  - Cray XC40 supercomputer
    - 2x Intel Xeon E5-2690 v3 CPU @ 2.60 GHz
    - Memory: 128 GB
    - Interconnection: Cray's Aries
- Software:
  - PCJ 4.1.0 / 5.0.6
  - Oracle JVM (HotSpot) 1.8.0
  - NCBI-BLAST 2.2.28+ / 2.3.0+

Results

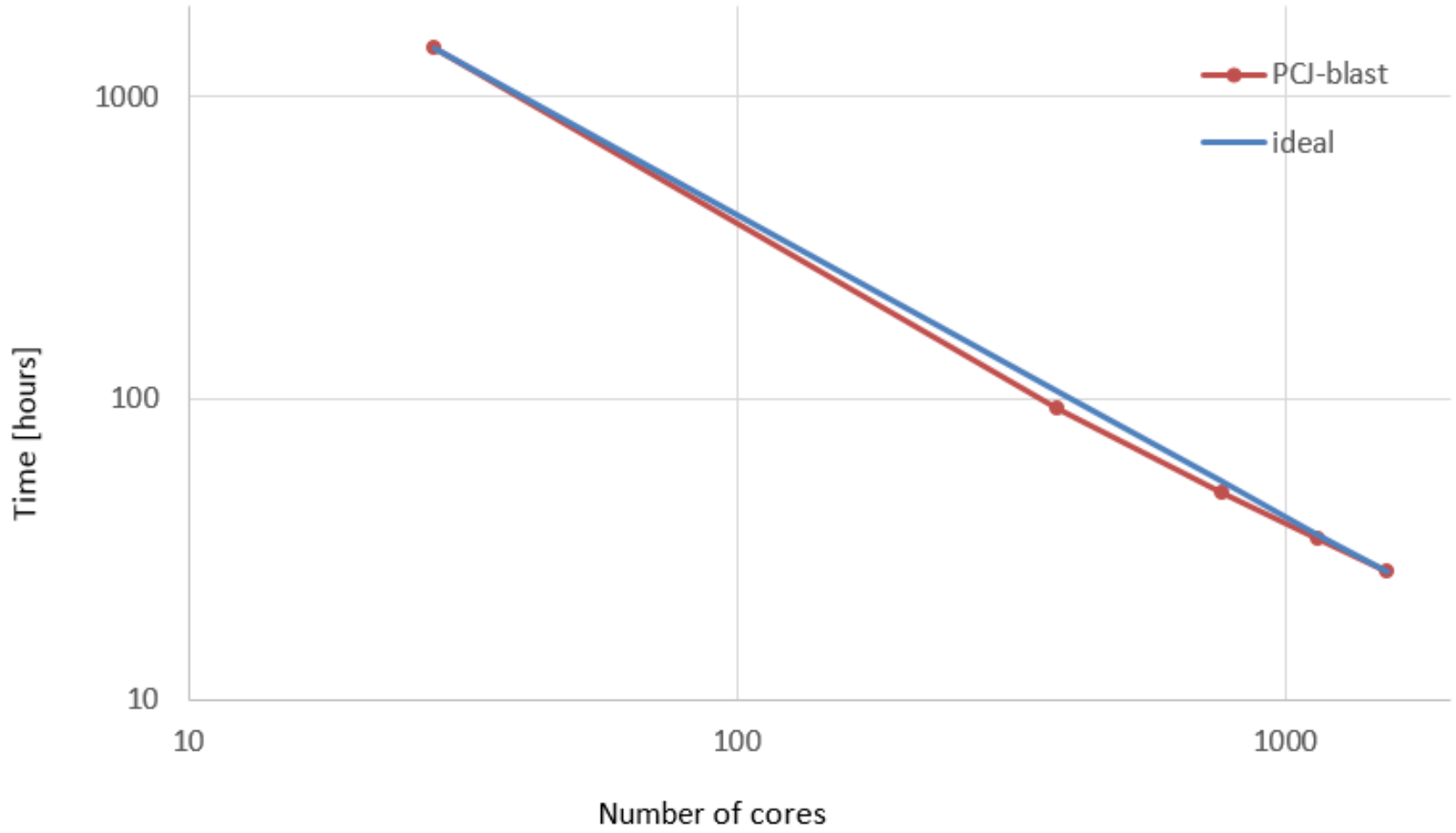
- The NCBI-BLAST can be executed with the `-num_threads` option
- Selected values:
  - for 28 core nodes (HPC x86 cluster): 4 PCJ threads with `-num_threads=7`
  - for 48 core nodes (Cray XC40): 4 PCJ threads with `-num_threads=12`

Processing 48 sequences at once using 1 instance of different version of NCBI-BLAST with different number of BLAST threads [Cray XC40]

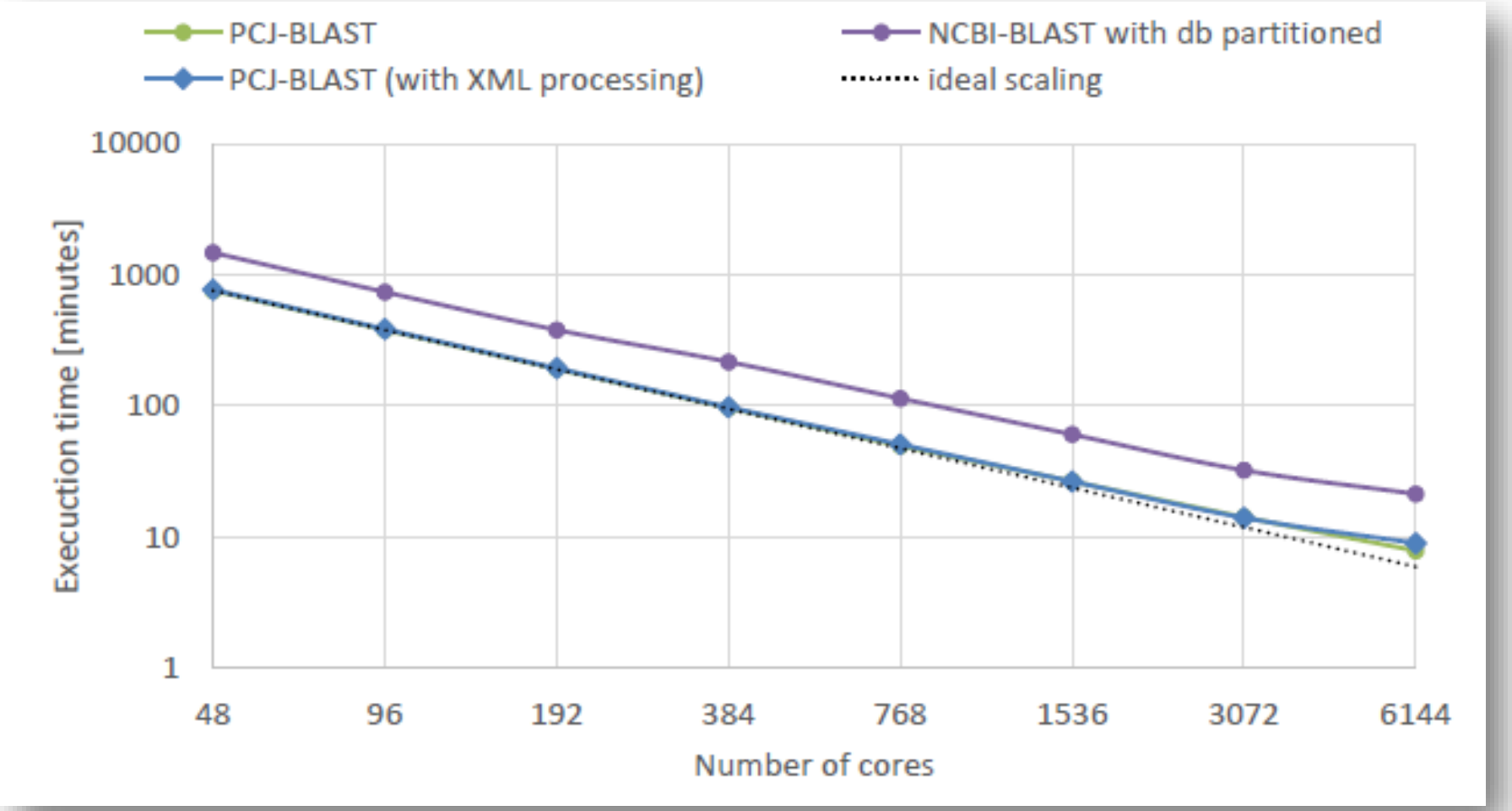


Results

Performance on HPC cluster [64 nodes : 1792 cores]



PCJ-BLAST - J. Comput. Biol. 2018 Aug;25(8):871-881



- DNA sequenca alignment
- PCJ-Blast – wrapper to NCBI-BLAST
- Performs dynamic load-balancing
- 2x faster than partition of the query and database

Conclusions

- Load balancing has been ensured by monitoring the execution of BLAST instances
- Application scale almost linearly
  - over 90% efficiency for 32 nodes (HPC cluster, Cray XC40)
  - 75% efficiency for 128 nodes (Cray XC40)
- Using 64 nodes instead of single node reduced the analysis time almost 55 times (from 1453h to 26.5h)
- The design and implementation were fast and efficient using the PCJ library
  - resulted in the preparation of the scalable application in short time

Piotr Bała bala@icm.edu.pl http://**pcj.icm.edu.pl**

PCJ development:

Marek Nowicki (WMiI UMK)
Michał Szynkiewicz (UMK, ICM UW) – fault tolerance

PCJ-Blast Tests, examples:

D. Bzhalava (Karolinska Instituet)

Acknowledgments:

NCN 2014/14/Z/ST6/00007 (CHIST-ERA grant),
EuroLab-4-HPC,
PRACE for awarding access to resource HazelHen at HLRS,
ICM UW (GB65-15, GA69-19)

Thank you!

http://pcj.icm.edu.pl

e-mail: bala@icm.edu.pl