POLYTECH

October 2017, St. Petersburg

SECR

Adapting Software Applications to Hybrid Supercomputer

Vsevolod Kotlyarov, Pavel Drobintsev,
Alexey Levchenko, Nikita Voinov

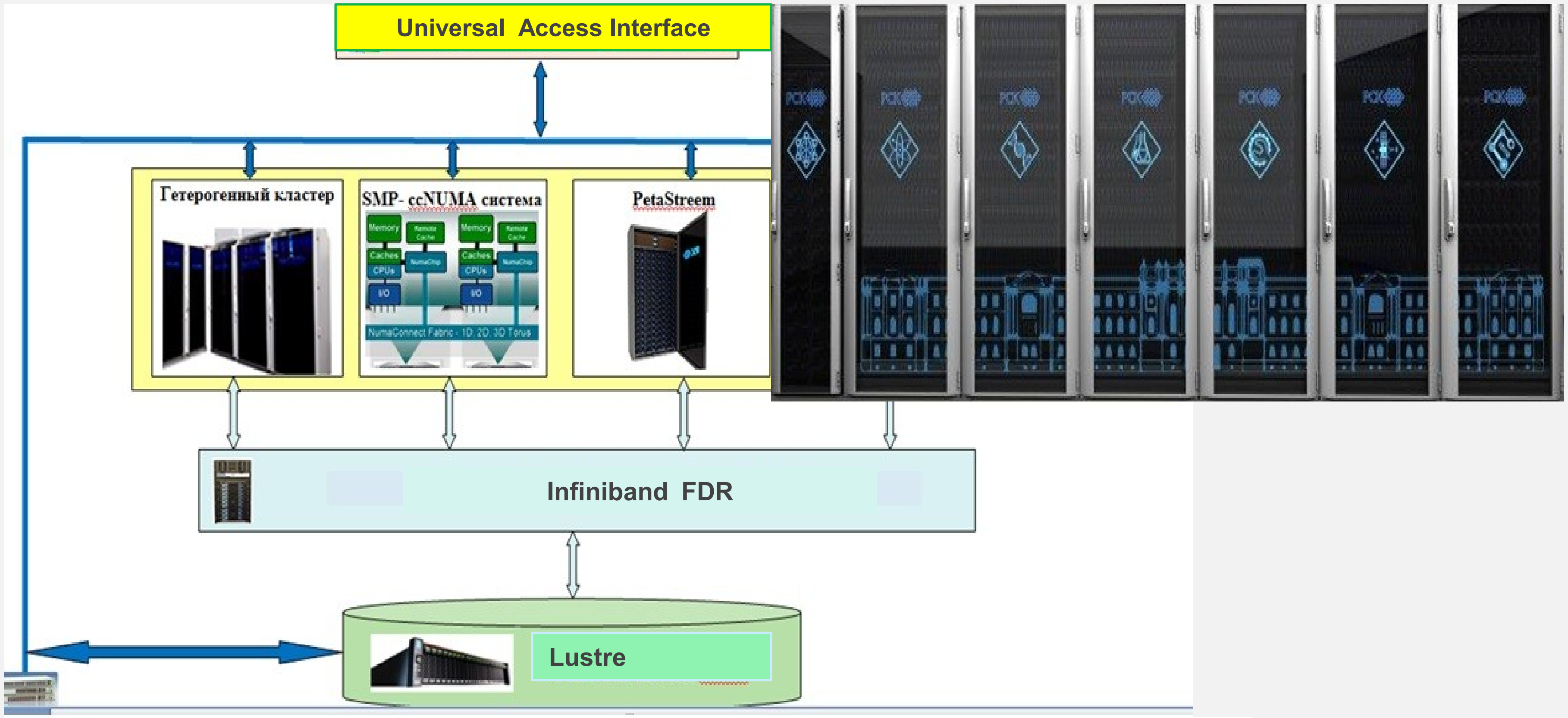
Problem relevance

- The work considers an approach to the adaptation of applications that use algorithms proven to be successful and developed with the help of computers that do not have a high degree of parallelism, though in a number of implementations they require a sharp reduction in the computing time. A natural way-out is to transfer the algorithm solution to a highly parallel heterogeneous processing environment, i.e. hybrid supercomputer. Unfortunately, the result does not always meet expectations

- The challenge is the need to consider architectural features of the supercomputer and the corresponding translation of the generic algorithm, while maintaining its semantic features, i. e. the development of parallel software of the generic algorithm scalable to allocated supercomputer resources. .

- Transformation to the parallel representation requires an analysis of dependencies in parallel threads on data and costs of the parallel supercomputer execution.

Features of SPbPU hybrid supercomputer



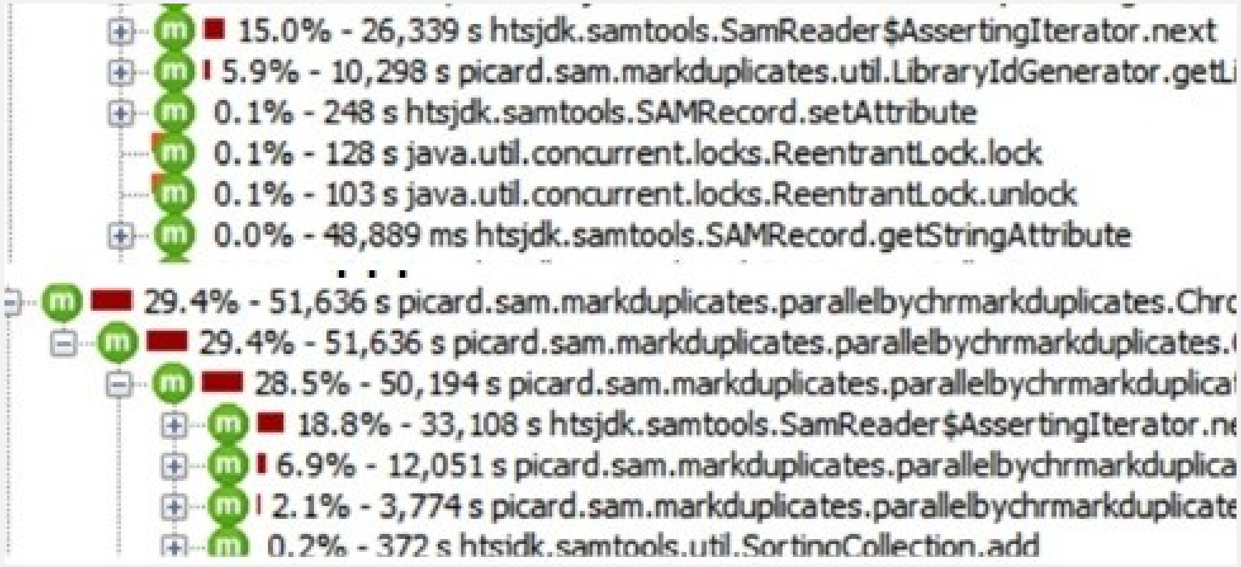
Tornado	18700 cor	42.7 TB	939 TFlops
NUMA	3000 cor	12 TB	31 TFlops
PETASTREM	15 300 cor	2 TB	259 TFlops

Problems of transformed algorithms

Following problems specific to the hybrid cluster application programming model shall be take into account into transformed algorithms:

- problem of the resource allocation dynamic control,
- problem of the task execution control in heterogeneous clusters,
- problem of the software adaptation for the effective execution in several clusters based on the required load.

Algorithm tree analysis

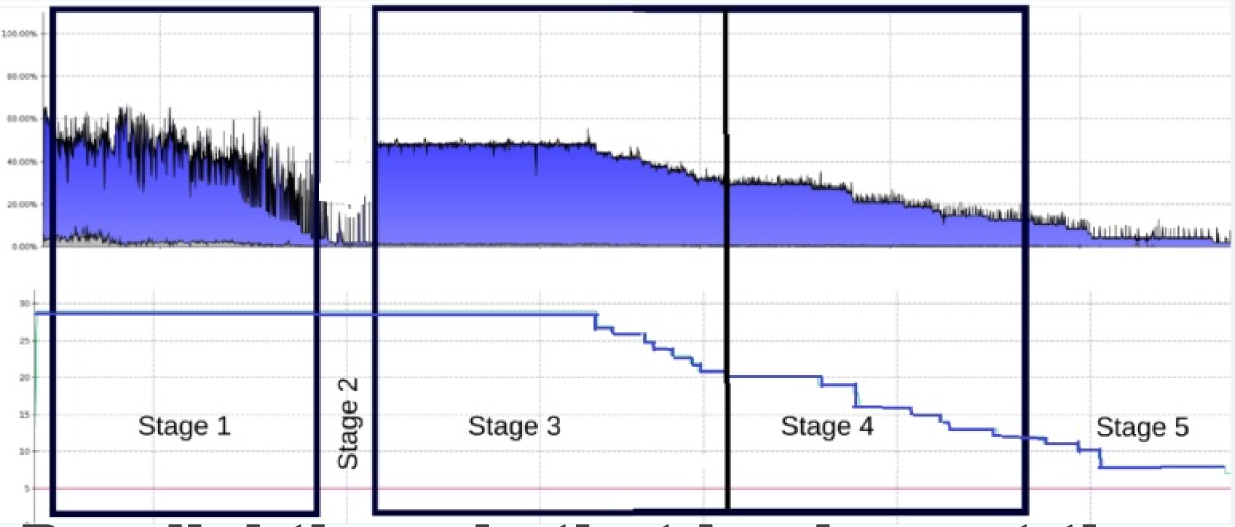


```
15.0% - 26,339 s htsjdk.samtools.SamReader$AssertingIterator.next
5.9% - 10,298 s picard.sam.markduplicates.util.LibraryIdGenerator.getLi
0.1% - 248 s htsjdk.samtools.SAMRecord.setAttribute
0.1% - 128 s java.util.concurrent.locks.ReentrantLock.lock
0.1% - 103 s java.util.concurrent.locks.ReentrantLock.unlock
0.0% - 48,889 ms htsjdk.samtools.SAMRecord.getStringAttribute
...
29.4% - 51,636 s picard.sam.markduplicates.parallelbychrmarkduplicates.Chrc
  29.4% - 51,636 s picard.sam.markduplicates.parallelbychrmarkduplicates.(
    28.5% - 50,194 s picard.sam.markduplicates.parallelbychrmarkduplicat
      18.8% - 33,108 s htsjdk.samtools.SamReader$AssertingIterator.ne
      6.9% - 12,051 s picard.sam.markduplicates.parallelbychrmarkduplica
      2.1% - 3,774 s picard.sam.markduplicates.parallelbychrmarkduplicate
      0.2% - 372 s htsjdk.samtools.util.SortingCollection.add
```

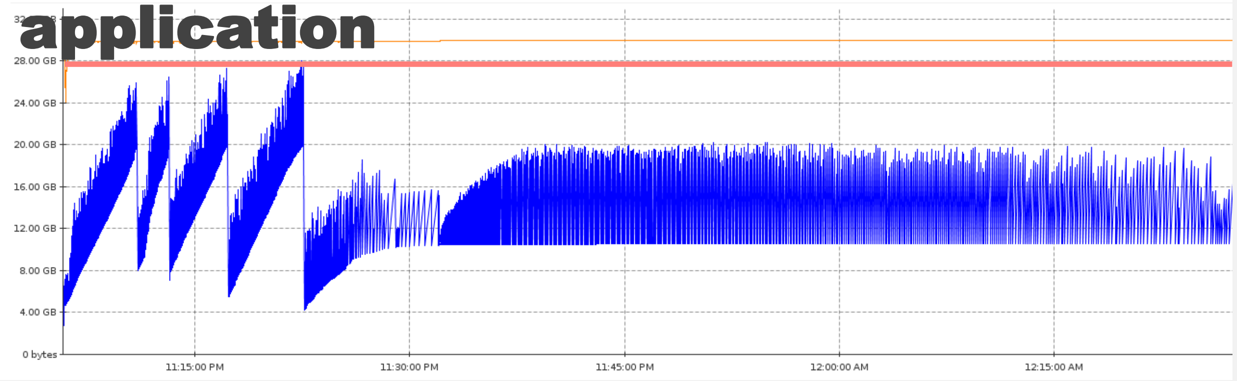
The result of the tree analysis, which is the identification of a list of methods with the highest frequency of calls

The algorithm correction through (initial or additional) parallelization by commonly used methods. During parallelization, dependencies in terms of data and control in parallel threads shall be considered with the provision of the proper synchronization.

Analysis of parallel threads and assessment of the thread-required memory space



Parallel threads that implement the application

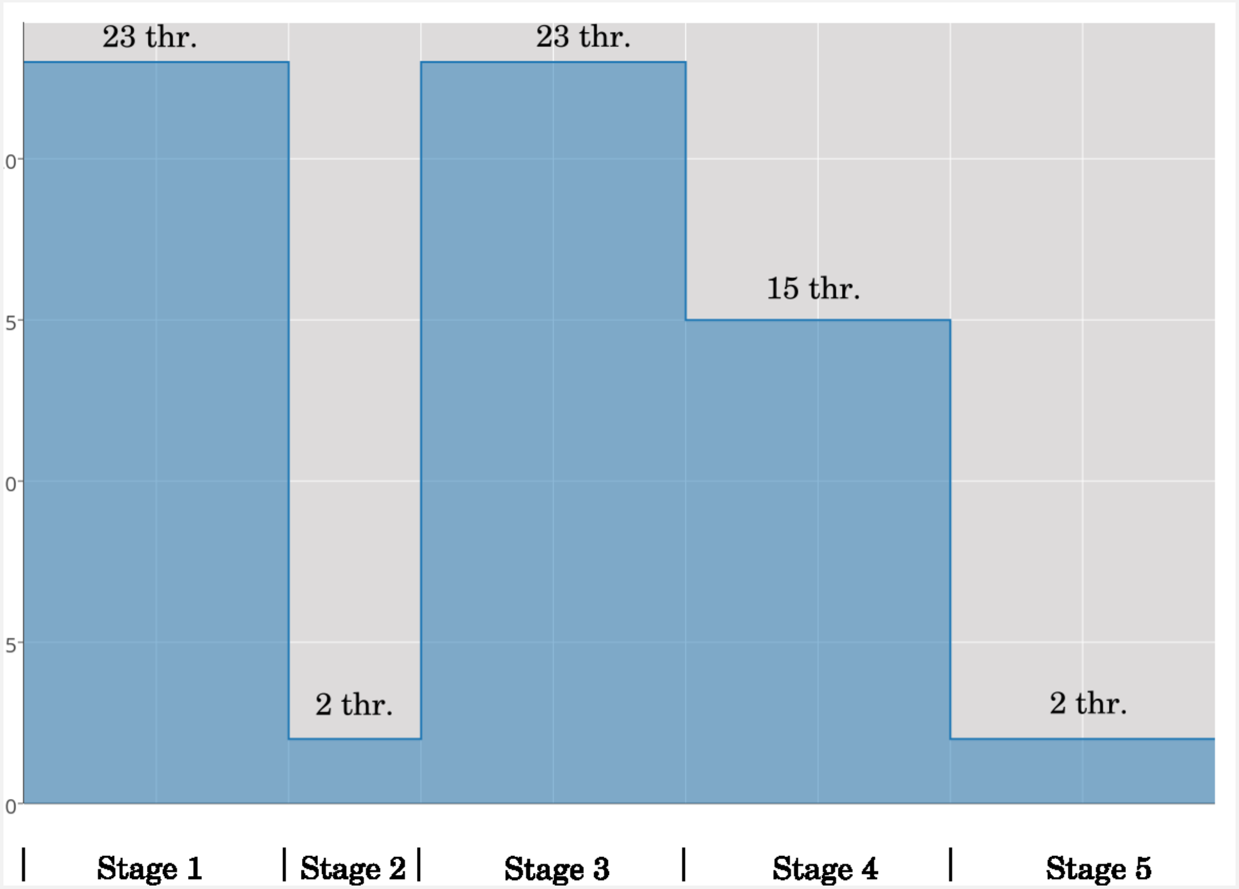


Requirements for the tread memory when executing the application

Indices of CPU total usage and the number of active threads shown in the diagram are divided into five areas, whose needs vary from 28 threads

if during the execution the thread-required memory space exceeds the allocated resource, for Heap or CASH are generated requirements for the additional algorithm fragmentation of each thread to fit into allocated resources.

Creation of an implementation application model for the supercomputer

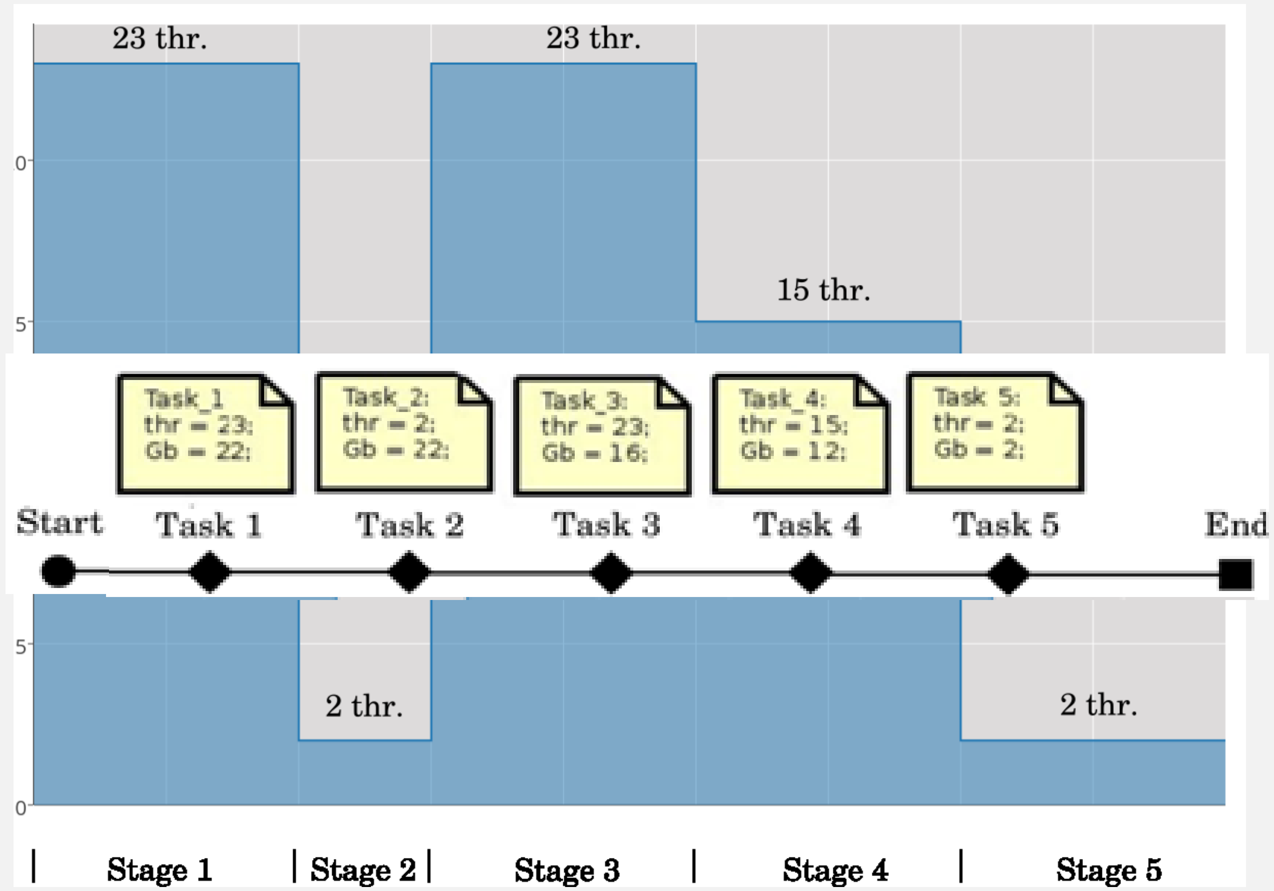


Model of application requirements for resources

The model takes into account requirements of the effective implementation of a particular application.

Model shall be transformed into a schedule for its implementation in the supercomputer.

Creation of an implementation application model for the supercomputer

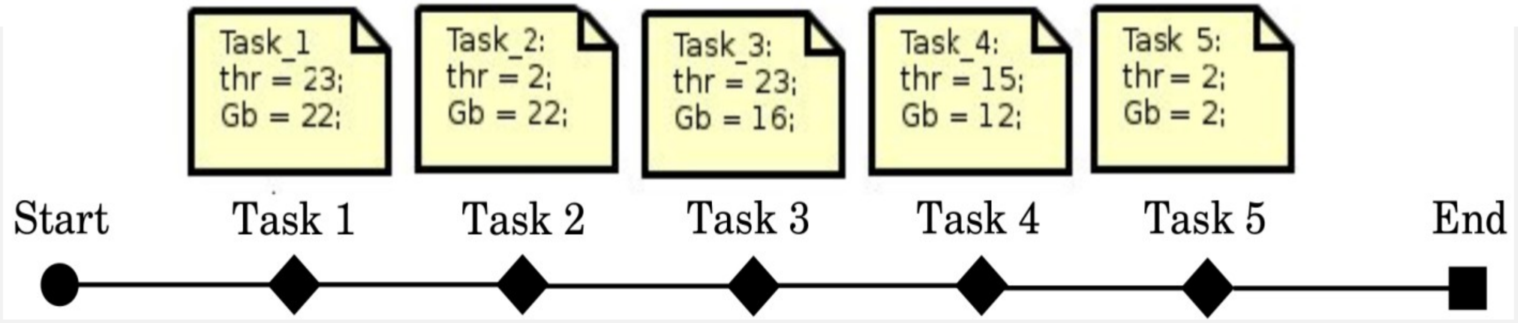


Model of application requirements for resources

The model shall be transformed into a schedule for its implementation in the supercomputer.

In this event, the application is transformed into a task package, each of which carries information about required resource

UCM language for tasks control scenario specification



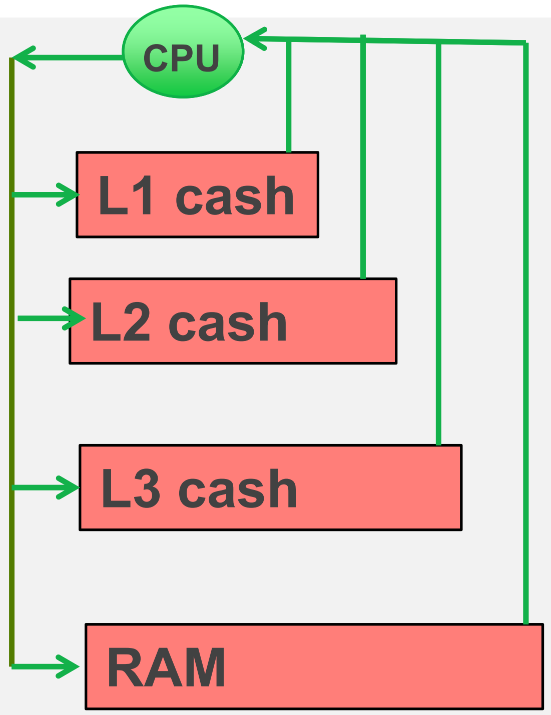
- The schedule formalization can be carried out in a number of ways. The work gives preference to the use of standardized control language UCM. The language provides control scripts in the form of flow graphs loaded with information about requirements for resources. Each node of the control graph is associated with a specific task determined through the analysis of the application solution algorithm. The UCM language is well-adapted for description of successive control scenarios, scenarios with alternatives and scenarios with iterations are used so far for the control description

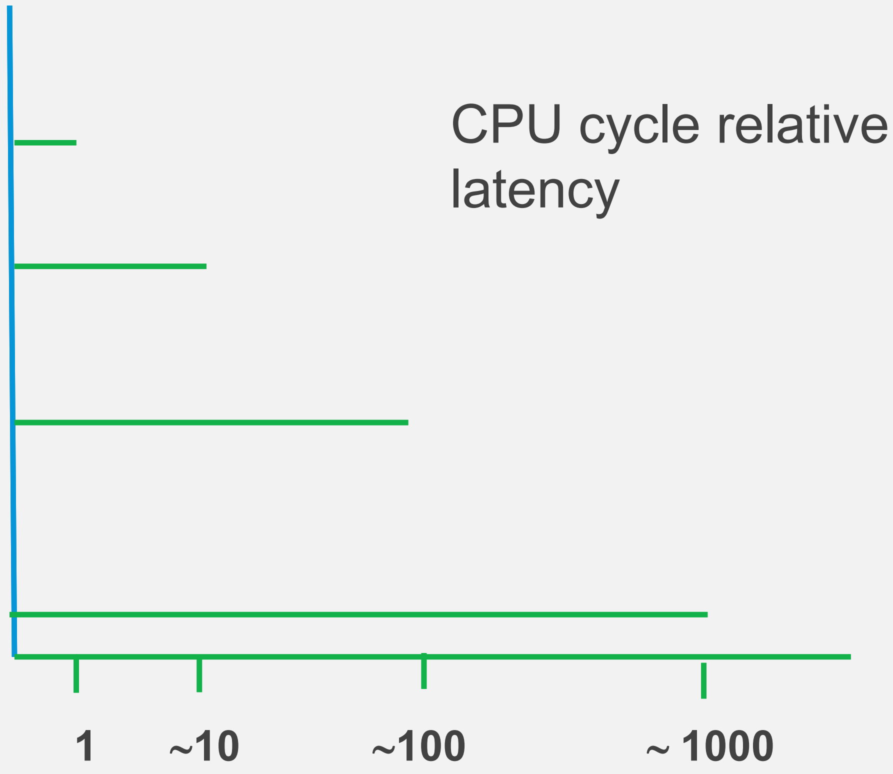
Example of the technique implementation for Big Data analysis algorithm

MarkDuplicates versions	Average Time	Max. Time
Scalable version, data 234Gb (1 cluster)	298 min	302 min
Scalable version, data 234Gb (2 cluster)	110 min	114 min
Linear version, data 234Gb (1 cluster)	1123 min	1134 min

The result is the algorithm executing acceleration by one order
But more is needed -> 2 order

Future: Threads analysis for mainly cash usage





Future research suggests refined analysis of required thread resources for algorithm transformation by dividing it into parts with resources are sufficient for executing in cash
A convenient tools for such transformation is OpenMP pragmas.

Conclusion

- The scope of the proposed technique application is, in the first place, processing algorithms for large data because the analysis of algorithms is usually costly and not always be fully automate. Therefore, the application of the technique is justified for repeatedly-used algorithms.
- For example, for algorithms of genetic research that should be quickly carried out in the medical practice. The practice of real implementation is so far limited algorithms in C++ and Java, for which there are sufficiently powerful profilers. Nevertheless, it shows good results in the adaptation of algorithms created regardless of solution specifics in a highly parallel supercomputer environment.
- The main drawback of the technique is its labor-intensive implementation. Therefore, it is planned in the future
  - to integrate of analysis tools for to facilitate better parallelization;
  - to automate the generation algorithm of schedules based on architectural features and resources of the execution environment to create a single in-process chain and appropriate tools to the adaptation of algorithms and for the efficient execution in hybrid supercomputer.

THANK YOU FOR ATTENTION

vpk@spbstu.ru