Numerical algorithms and fault-tolerance of hyperexascale computer systems

Boris Chetverushkin, Marina Kornilina, ***Mikhail Yakobovskiy***

lira@imamod.ru

03-04 May 2018

Progress in HPC performance

- 1990-2000s – Giga : 10^9 FLOPs
- 2000s – Tera : 10^{12} FLOPs (first announced 1TFLOPs- 1994)
- 2007-2008 – First PetaFLOPs systems : 10^{15} FLOPs
- 2016-2017 – Wide use of PetaFLOPs computer systems
- 2019…2020… – Expectations of building and exploiting ExaScale computing systems capable of 1 ExaFLOPs and more : 10^{18} FLOPs

Challenges in exploitation of HPC:

- Resiliency
- Efficient use of full computing power (shortage of models, numerical algorithms and program tools)
- Demand for adequate logically simple though efficient algorithms

- ExaScale computer

Peak speed of Exaflops is 10^{18} FLOPs flops = 1,000 Petaflops

"Exascale" is based on the ratio of execution time of full applications

– can run real applications
50X faster than it runs on Titan, Sequoia (20PF systems),
or 100X faster than on Mira (10 PF),
or solve more complex problems

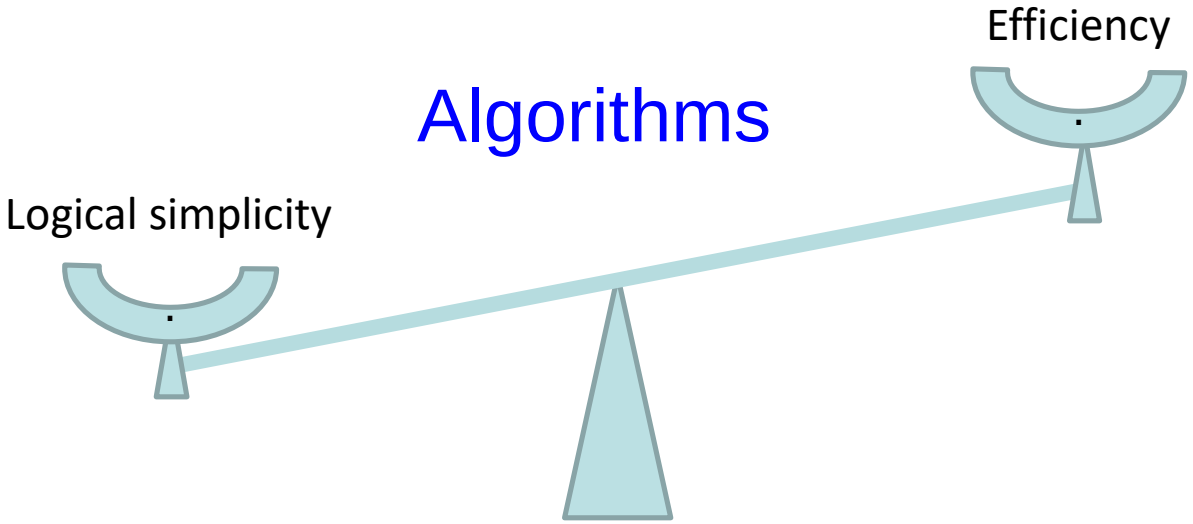
– operates in a power envelope of 20-30 MW

– is sufficiently resilient (average fault rate due to hardware or system faults is once a week, or less)

– has a software stack that supports a broad spectrum of applications and workloads.

Paul Messina, Argonne National Laboratory, ECP Director
"A Path to Capable Exascale Computing", July 31, 2016

Algorithms



- Explicit schemes allow to create a logically simple algorithms, but they have the strict restrictions on the time step in the stability conditions:
  - Stability condition for hyperbolic equations:

    $$\Delta t \le h,$$

    where Δt *is a time step, and* h *is a space step*
  - Stability condition for parabolic equations:

    $$\Delta t \le h^2$$

This condition is an obstacle to using high space resolution

Poisson equation (gravity potential)

$$\frac{\delta^2 \varphi}{\delta x_i^2} = 4\pi G \rho(r)$$

$$\rho(r) = \begin{cases} \rho & r \le R \\ \delta & r > R \end{cases}$$

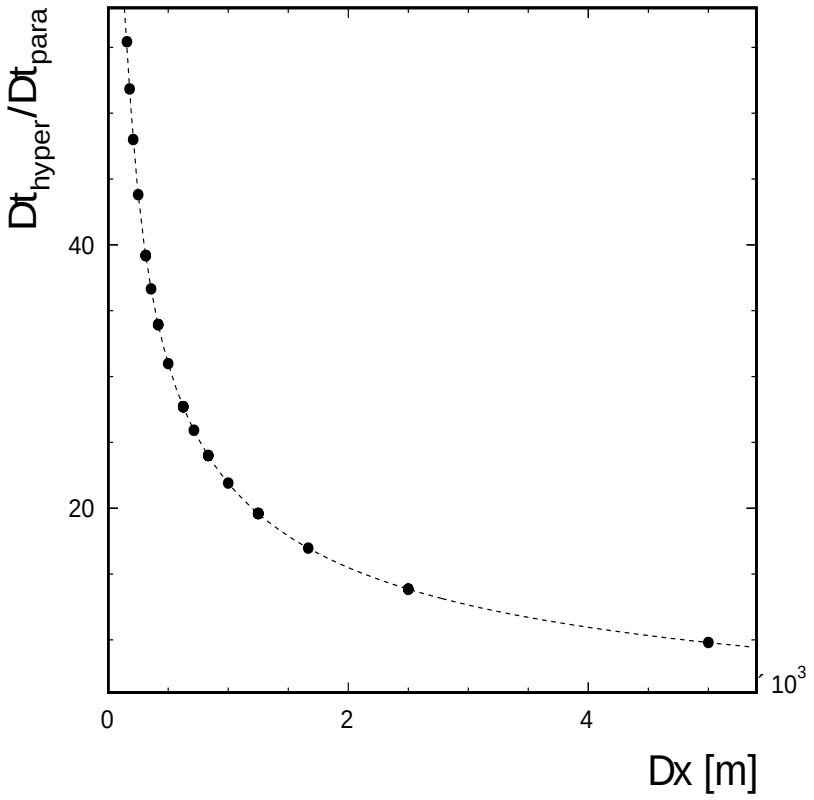
$$\frac{\varphi_i^{j+1} - \varphi_i^j}{\Delta t} = (\varphi_{\bar{x}})_x^j + F$$

$$\frac{\varphi_i^{j+1} - \varphi_i^{j-1}}{2\Delta t} + \tau^* \frac{\varphi_i^{j+1} - 2\varphi_i^j + \varphi_i^{j-1}}{\Delta t^2} = (\varphi_{\bar{x}})_x^j + F$$

We can transform the equation by adding appropriate small term of the order of the second time derivative and get hyperbolic equation with soft stability condition

Poisson equation (gravity potential)

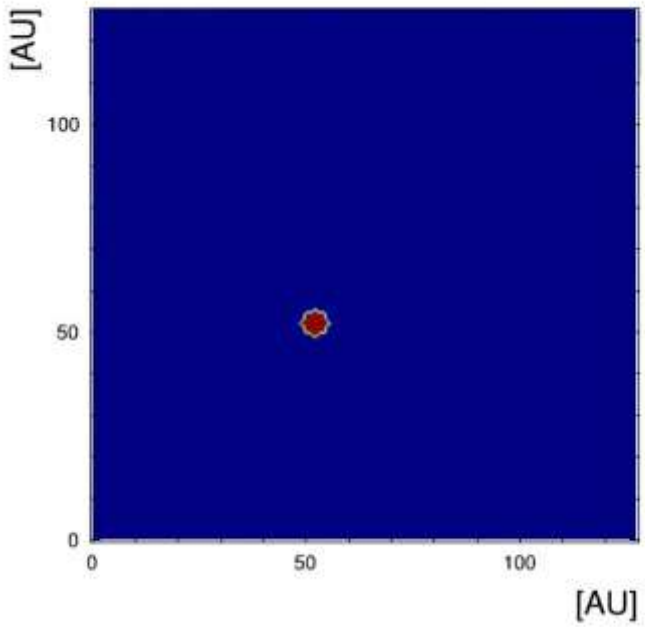
The ratio of the time steps Δt of parabolic and hyperbolic method as a function of space step Δx



The smaller Δx the greater the gain in the time step achieved due to the hyperbolization

Accretion of a cloud of interstellar gas on a compact astronomical object

Low resolution



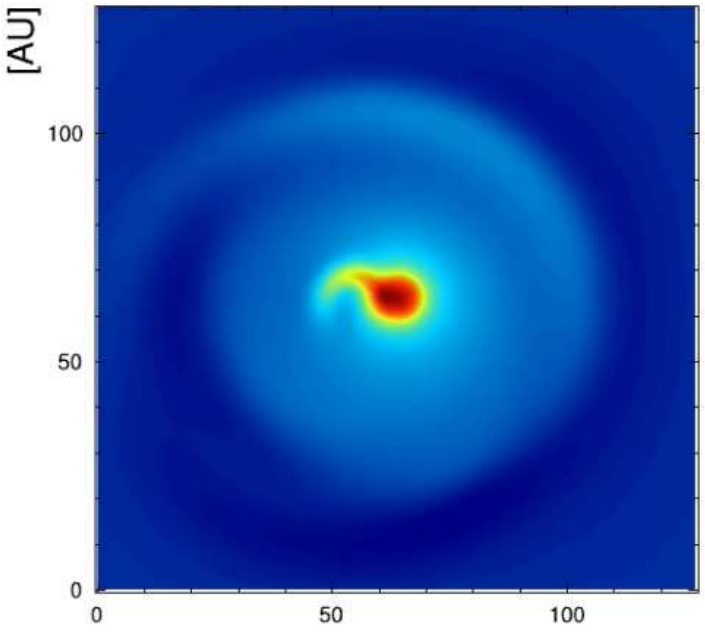
High resolution

- Interstellar cloud 5 AU
- Density 0.8 x 10^{-11} кг/m^3
- Cloud speed 300 m/s
- Impact parameter 4-10 AU

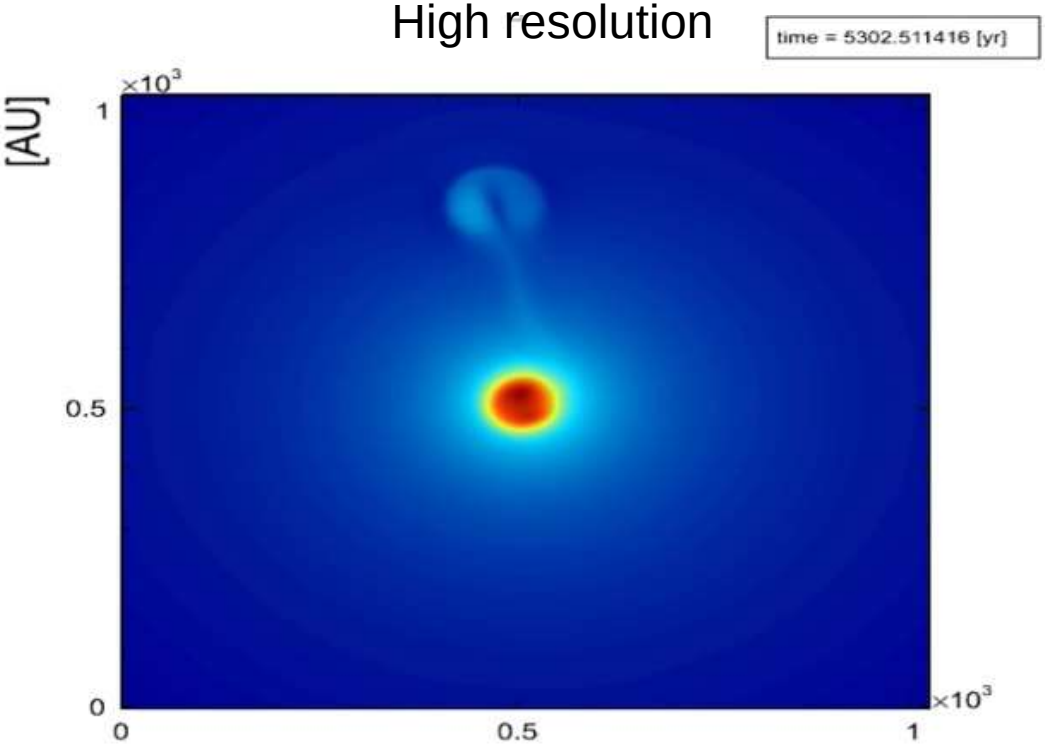
- compact astronomical object:
  - Weight 1030 Kg
  - Radius 0.5 AU
- Temperature of space T = 20 K

Accretion of a cloud of interstellar gas on a compact astronomical object

Low resolution



High resolution



- Interstellar cloud 5 AU
- Density 0.8 x 10^{-11} кг/m^3
- Cloud speed 300 m/s
- Impact parameter 4-10 AU

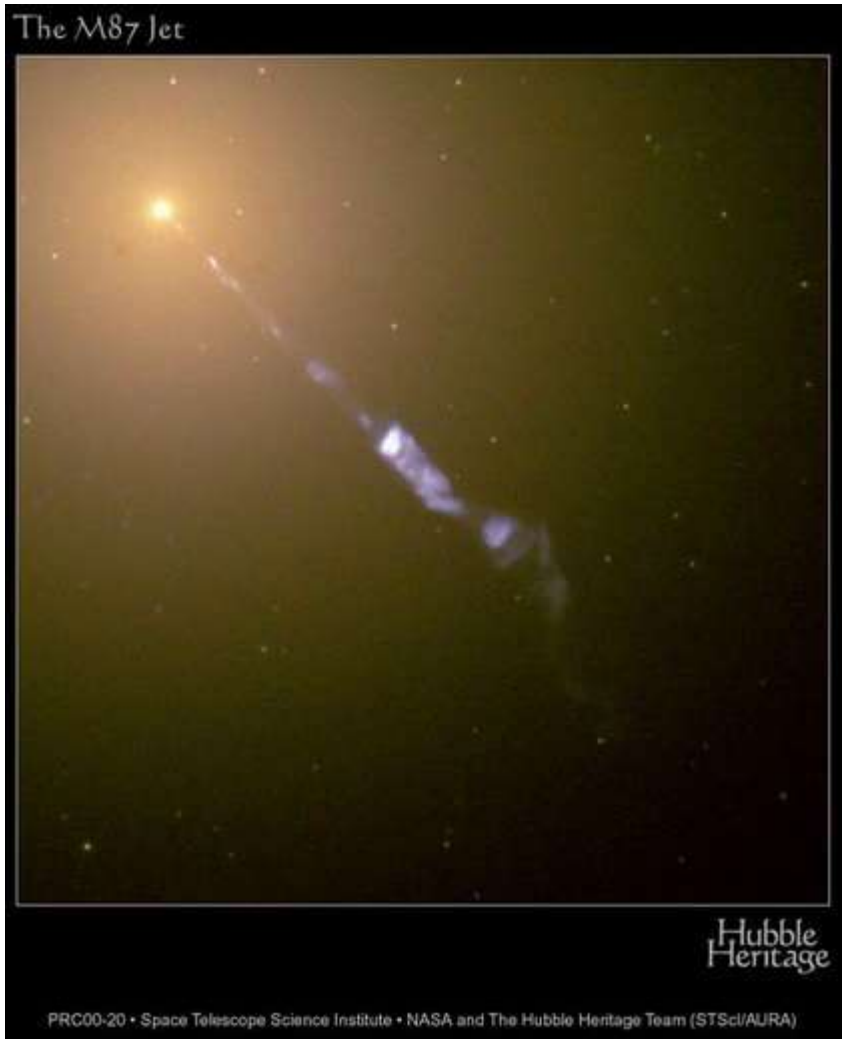
- compact astronomical object:
  - Weight 1030 Kg
  - Radius 0.5 AU
- Temperature of space T = 20 K

A Jet from Galaxy M87

The most popular hypothesis holds that the jet is created by energetic gas swirling around a massive black hole at the galaxy's center

Hubble Space Telescope

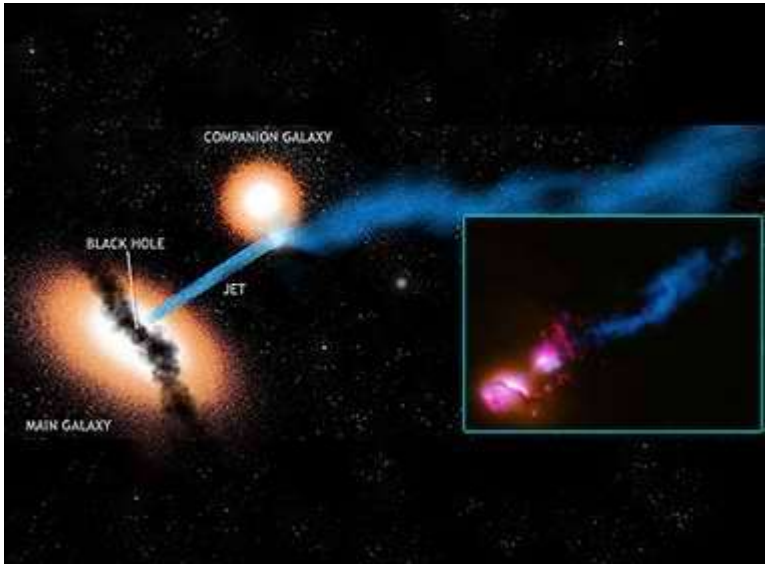
Astronomy Picture of the Day, July 6, 2000
https://apod.nasa.gov/apod/ap000706.html



http://planetarium-kharkov.org/?q=galaxy-CGCG049-033

A powerful jet from a super massive black hole is blasting a nearby galaxy

This is a system of two galaxies C321



https://ufn.ru/ru/news/2008/1/
https://phys.org/news/2007-12-death-star-galaxy-black-hole.html

http://planetarium-kharkov.org/?q=galaxy-CGCG049-033

Blue Waters system

Study on Blue Waters (Cray HLRS – Germany, Stuttgart) showed

- an event that required remedial repair action occurred on average every 4.2 hours
- system-wide events occurred approximately every 160 hours.

Di Martino, Catello, Zbigniew Kalbarczyk, Ravishankar K. Iyer, Fabio Baccanico, Joshi Fullop, and William Kramer. "Lessons learned from the analysis of system failures at petascale: The case of blue waters." In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pp. 610-621. IEEE, 2014.

Reliability of HPC

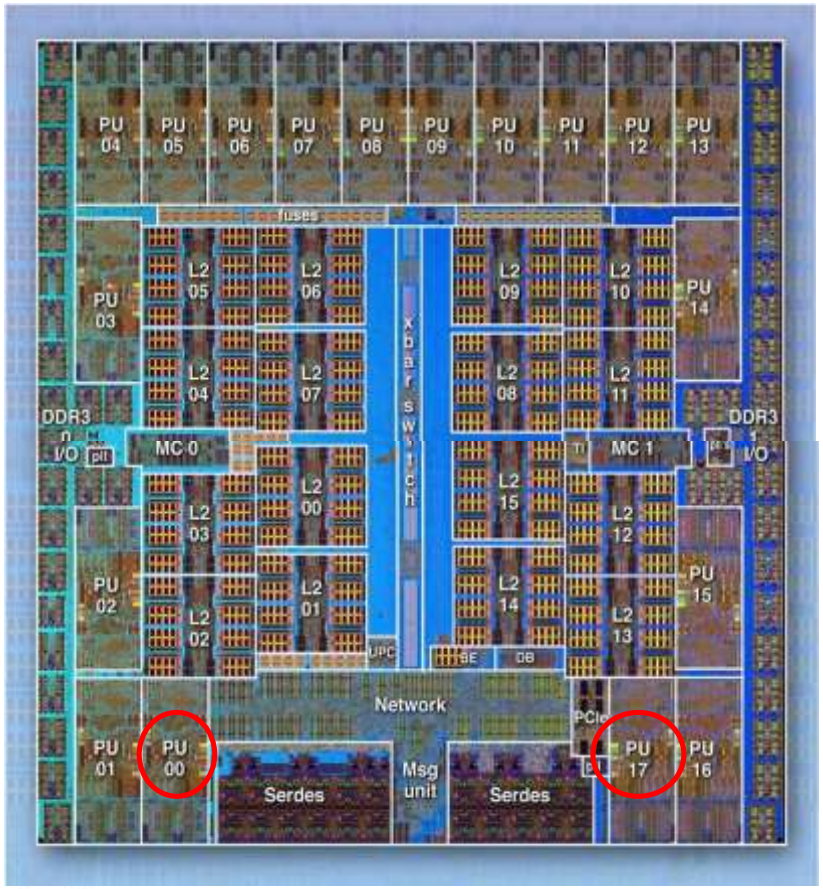
System	CPUs	Reliability
ASCI Q	8192	MTBF (Mean time between failures): 6.5 hours Leading outage sources: storage, CPU, memory
ASCI White	8192	MTBF: 5 5.0 hours (‘01) and 40 hours (‘03). Leading outage sources: storage, CPU, 3rd-party HW.
PSC Lemieux	3016	MTBI: 9.7 hours.
Google(as of 2003)	15000	20 reboots/day; 2-3% machines replaced/year. HW outage sources: storage, memory.

D. Reed. High-end computing: The challenge of scale. Director’s Colloquium, Los Alamos National Laboratory, May 2004.

A study conducted by LANL in 2005 estimated the MTBF to be 1.25 hours on a petaflop machine.

Philp. Software failures and the road to a petaflop machine. In HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of HPCA-11. IEEE Computer Society, 2005

IBM PowerPC® A2 1.6 GHz, 16 cores per node



Robert W. Wisniewski.

BlueGene/Q: Architecture,

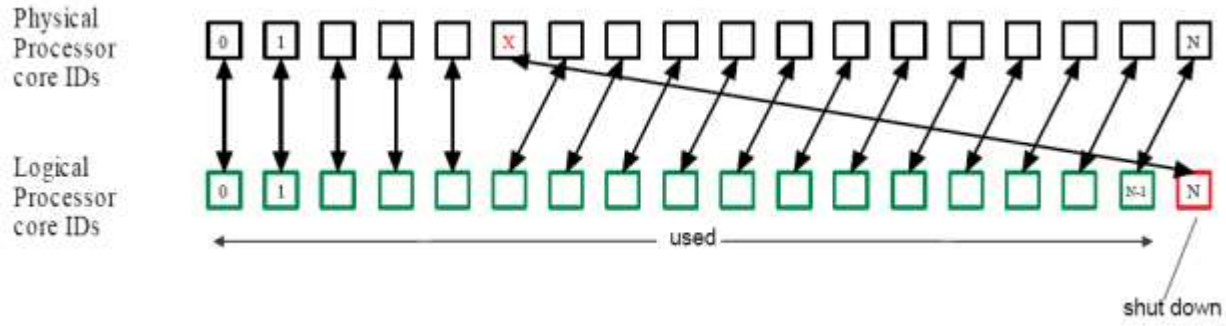
CoDesign; Path to Exascale / Blue

Gene Supercomputer Research,

January 25, 2012

There are two spare cores here.
One core performs service functions.
One core is idle.

Cores are renumbered.
The idle one includes into work.



Exascale resilience

Hardware faults are expected to be more frequent :

- The smaller transistors are more error prone because of cosmic radiation
- The smaller circuits are more easily upset because they carry smaller charges

Software will be more complex and hence more error-prone:

- As hardware becomes more complex (heterogeneous cores, deep memory hierarchies, complex topologies, etc.), system software becomes considerably more complex.
- Multiphysics and multiscale codes couple an increasingly large number of distinct modules. The need to reduce communication, allow asynchrony, and tolerate failure results in more complex application codes.

Cappello, F., Geist, A., Gropp, W., Kale, S., Kramer, B., & Snir, M. (2014). Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, *1*(1).

Prediction

- Researchers predict that large parallel applications may suffer from different faults as frequently as once every 30 minutes on exascale platforms

(Marc Snir, et al. Addressing failures in exascale computing. International Journal of High Performance Computing Applications, 28(2):129–173, May 2014.)

- Fault tolerant technique is essential for HPCs
- The goal is to develop a fault tolerant technique suitable for long-lasting numerical simulations on Exascale Systems.

The main strategy is reserving

Fortunately, we do not need complete fault-tolerance.

It is enough to have a computer system which provides correct work within a required time interval.

This may be achieved by using of excessive

- elements in structure (processors, memory)
- data (ECC, messages)
- periodical reserving of data (checkpointing)
  - ? global or partial
  - ? how often
  - ? Where (shared or centralized)

Time to create a checkpoint in distributed file system
~ 30 min

System from TOP 500	Max performance	Checkpoint time (minutes)
LLNL Zeus Lawrence Livermore National Laboratory	11 TeraFLOPS	26
LLNL BlueGene/L	500 TeraFLOPS	20
Argonne BlueGene/P	500 TeraFLOPS	30
LANL RoadRunner **Los Alamos Natlional Labs**	1 PetaFLOPS	~ 20

Cappello, F. 2009. Fault Tolerance in Petascale/ Exascale Systems: Current Knowledge, Challenges and Research Opportunities. International Journal of High Performance Computing Applications 23, 3, 212–226.

Yet that MTBF for Exascale machines may be less than one hour.

Checkpointing level

- System level
  - Simplicity of use
    - But, we must recalculate all steps starting from the last checkpoint !
- User (application) level
  - Crucial reduction of amount of stored data is achieved
  - “Replacement” of compute node is used instead of task restart
  - Data storing is performed not only in local drives HDDs but also in RAMs

Fault-tolerant environments for checkpointing

Automatic (based on BLCR) system level checkpoint :

- MPICH, MVAPICH, OpenMPI

Semi-automatic, user level checkpoint :

- C^3 - Cornell Checkpoint pre-Compiler, (Greg Bronevetsky, Daniel Marques, …)
- ULFM(FT-MPI)

Egwutuoha, I.P. A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. / I.P. Egwutuoha, D. Levy, B. Selic, S. Chen // The Journal of Supercomputing. — 2013. — Vol. 65, No.3. —P. 1302-1326.

Cappello, F. Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities // International Journal of High Performance Computing Applications. — 2009. — Vol. 23, No. 3. — P. 212–226

ULFM - User-Level Failure Mitigation

Current MPI 3.1 itself provides no mechanisms for handling processor failures.

ULFM is designed according to be the minimal interface necessary to restore the complete MPI capability to transport messages after failures.

ULFM functions:

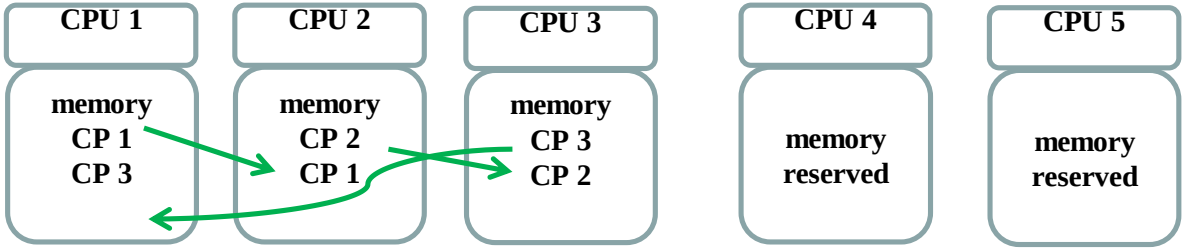
- **MPI_COMM_REVOKE**
- **MPI_COMM_SHRINK**
- **MPI_COMM_FAILURE_GET_ACKED**
- **MPI_COMM_FAILURE_ACK**
- **MPI_COMM_AGRE**

http://fault-tolerance.org/

ULMF is a part of new version of MPI (MPI 4.1)

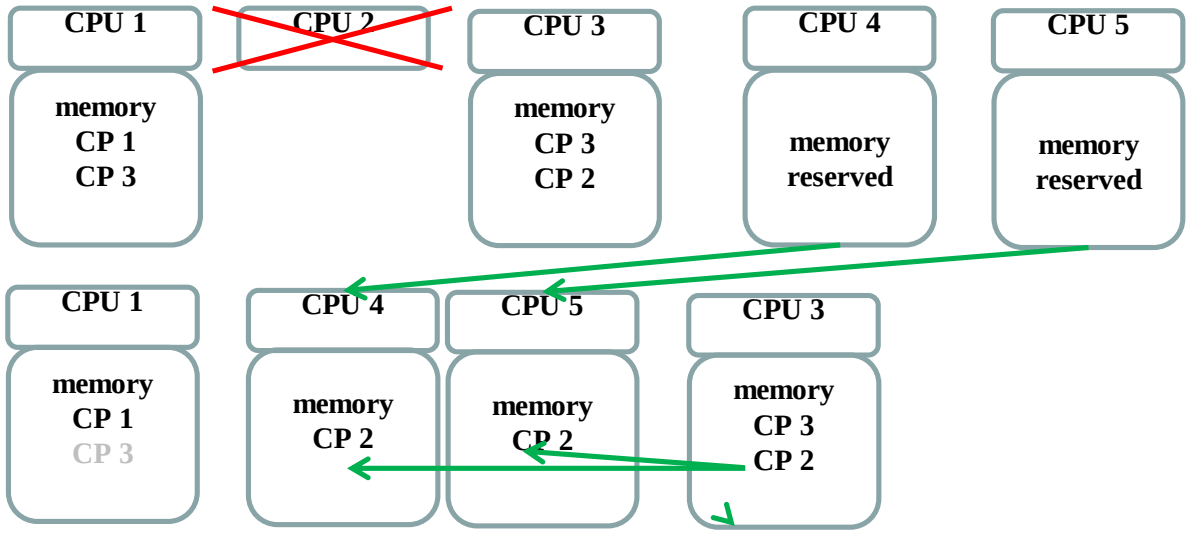
Fault tolerance approach

If we do not want to rollback we must have several copies.
We store them in the local memory of other working processors.



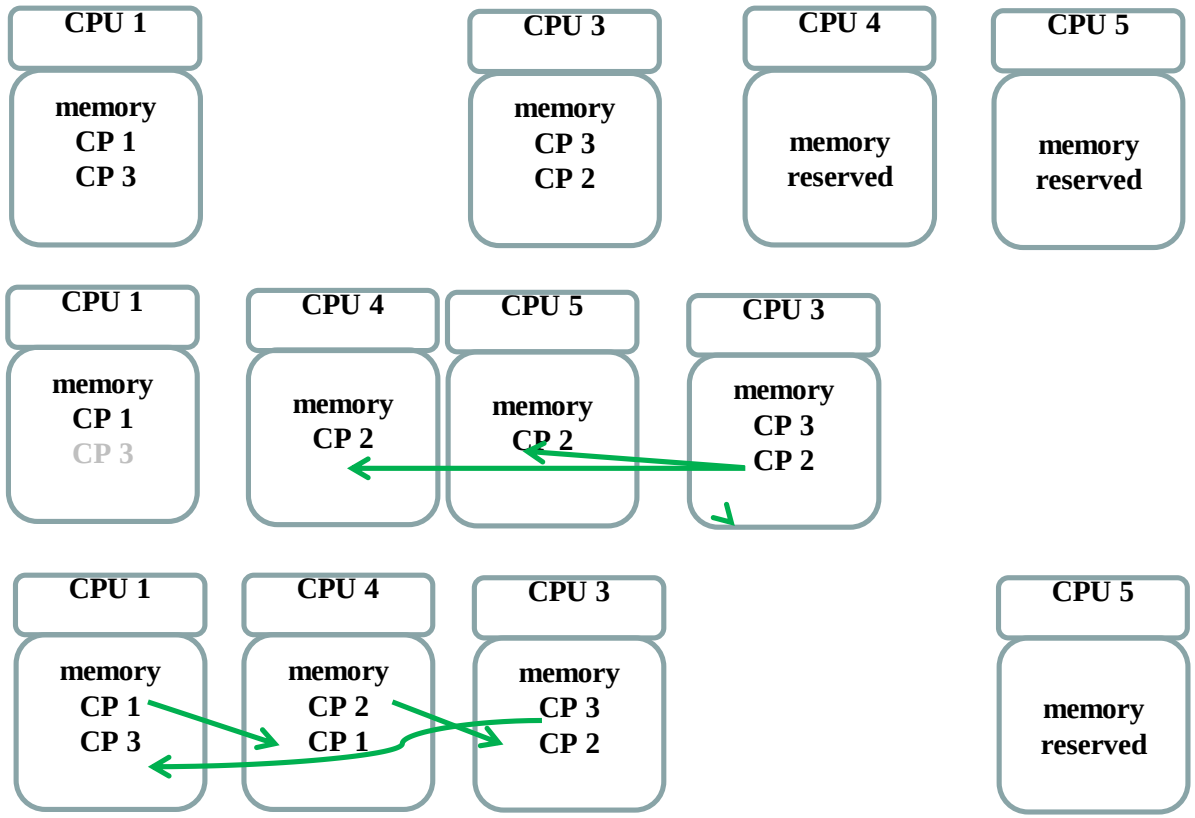
Fault tolerance approach

If we do not want to rollback we must have several copies.
We store them in the local memory of other working processors.



Fault tolerance approach

If we do not want to rollback we must have several copies.
We store them in the local memory of other working processors.

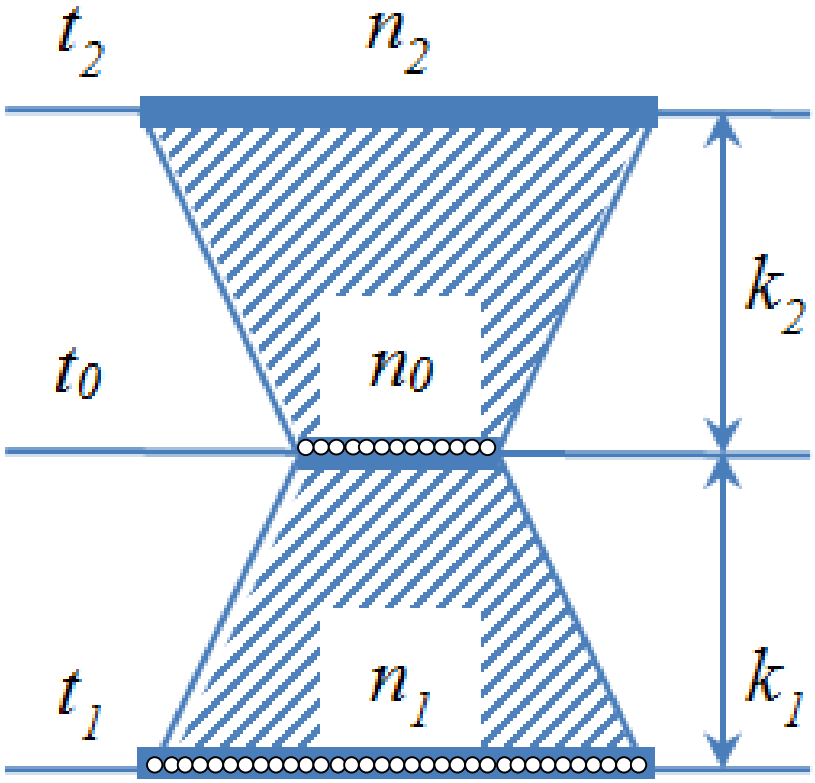


Hyperbolic equation1D

$$\frac{\partial^2 \Phi}{\partial x^2} - \frac{1}{C^2}\frac{\partial^2 \Phi}{\partial t^2} = F(x,t)$$

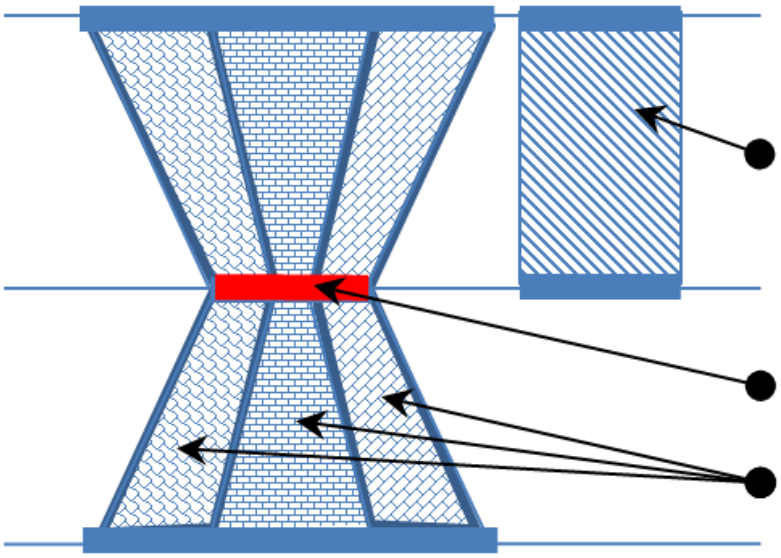
Two characteristic lines $x\text{-}ct=c_0$ and $x+ct=c_1$ define the area, which has influence on the solution $\Phi(x,t)$ in point (x,t).

The domain n_1 allows to recover the data lost due to a processor failure using accelerated recalculations



The data in the domain n_1 determine the solution $\Phi(x,t)$ in the domain n_0 at the time t_0.

Replace of one failed processor by three reserve processors



Complete recovery of computation process

Continuation of the calculations by operable processors

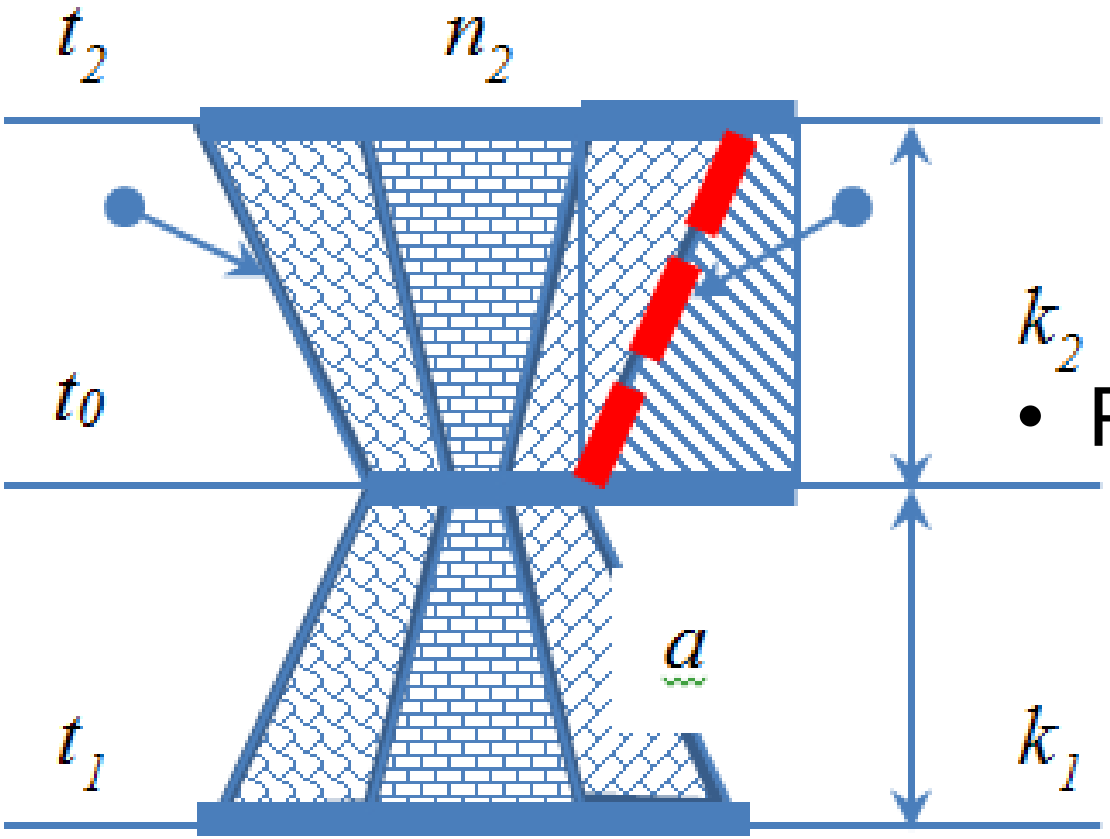
Processor failure

Lost part of the computational grid

Recalculation of lost data

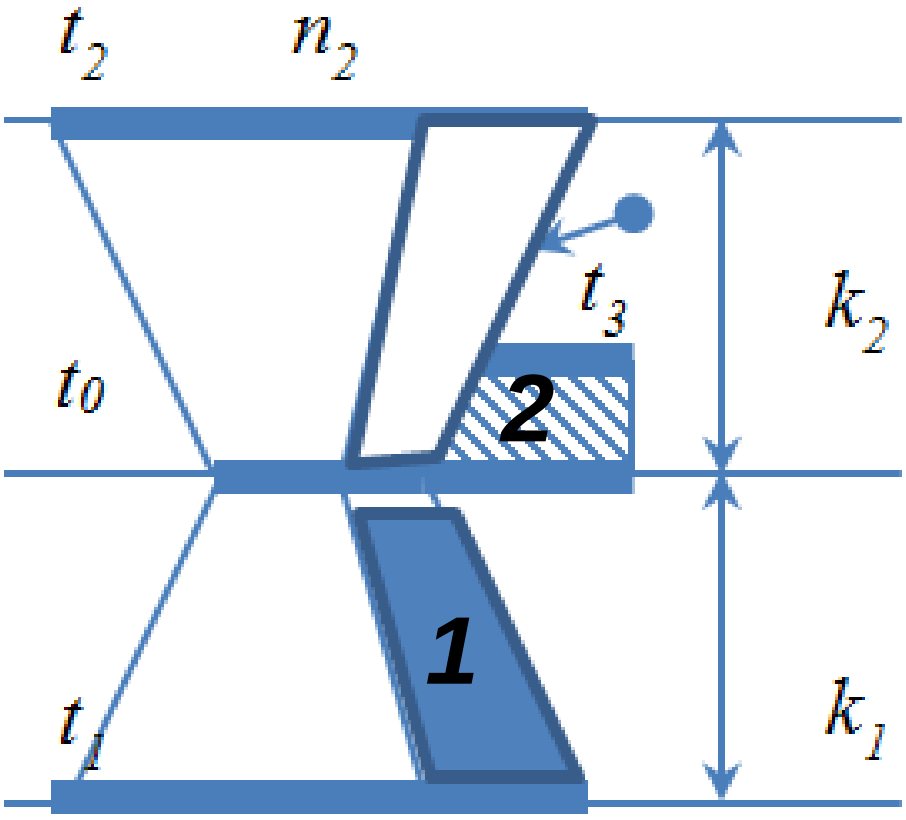
Local checkpointing record

Replacement of a single failed processor by three reserve processors



- Logging
- Recalculating

Simultaneous calculation of two domain fragments by two processors

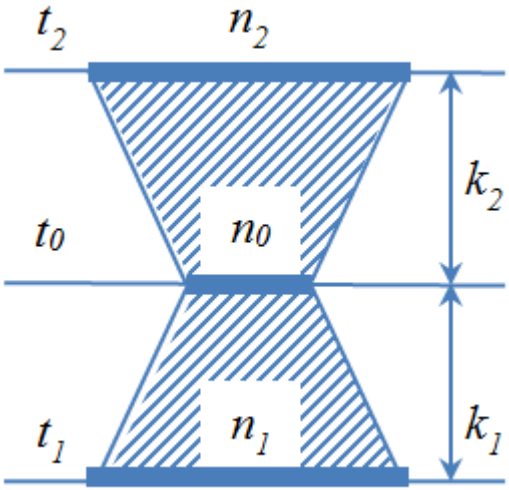


Estimate for the number of additional processors required for recalculation (single processor failure)

$$p_d > \frac{2}{d+1}\sum_{i=0}^{d}\alpha^i$$

$$\alpha = 1 + 2\gamma\frac{k_1}{n_0}$$

$$\gamma = \frac{C\Delta t}{h}$$ - Courant number



n_0^d – the number of calculation points initially processed by each processor

$d = 1, 2\ or\ 3$ – dimension of the simulated space

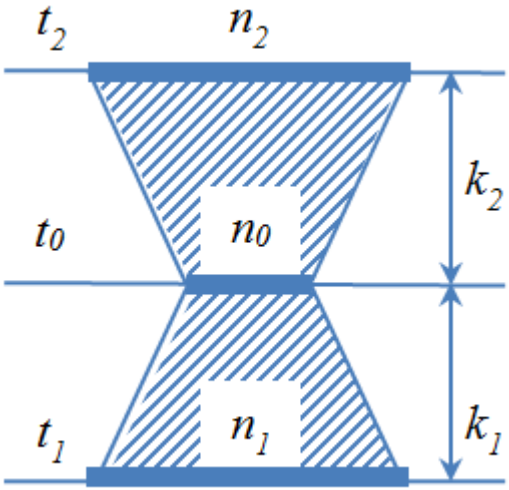
How much is α ?
Let the γ be equal to 1

$$p_d \geq \frac{2}{d+1}\sum_{i=0}^{d}\alpha^i$$

$$\alpha = 1 + 2\frac{k_1}{n_0}$$

$$n_0 = \sqrt[d]{n}$$

Let the $\alpha \approx 1$

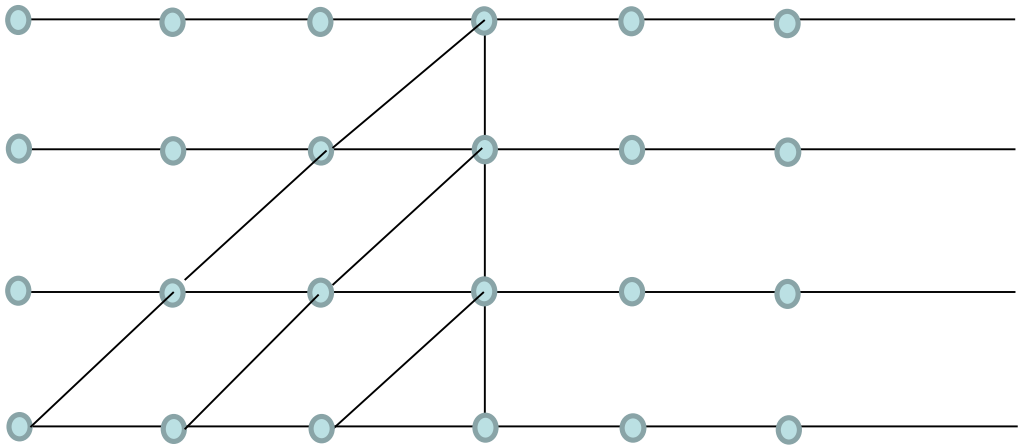


n_0^d – the number of calculation points initially processed by each processor

$d = 1, 2\ or\ 3$ – dimension of the simulated space

$\gamma = 1$ - identical result

- Only hyperbolic?
  - No, any explicit scheme



- Each step increases the zone of dependence by one cell

How many additional processors we need?

$$\gamma = 1$$

$$p_d \geq \frac{2}{d+1}\frac{\alpha^{d+1}-1}{\alpha-1} \qquad \alpha = 1 + 2\frac{k_1}{n_0}$$

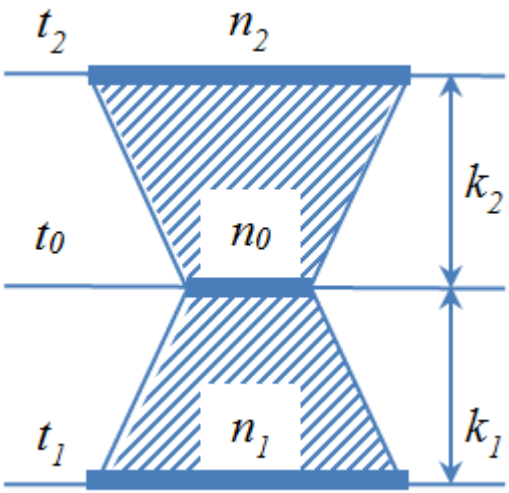
$$n_0 = \sqrt[d]{n}$$

$k_1 \ll n_0$??

$$\alpha \approx 1$$

$$p_d > 2\frac{\sum_{i=0}^{d}\alpha^i}{d+1}$$

$$p_d > 2$$

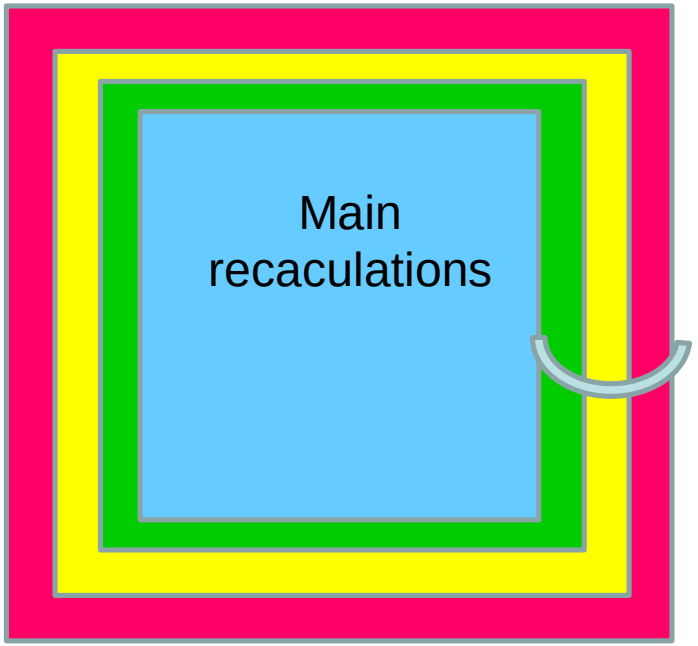


n_0^d – the number of calculation points initially processed by each processor

$d = 1, 2\ or\ 3$ – dimension of the simulated space

Additional part is only the border

$$p_d > 2$$



Conclusion

- A new method is suggested which provides the continuation of long-term calculations on a million core computing system in the presence of faults
- The method relies on the locality properties of hyperbolized systems of partial differential equations, for which the domain of dependence on the solution is localized in space
- The necessary part of the solution can be rapidly recalculated without rollbacking and restarting the whole calculation process
- Time required for recalculation is small in comparison to the time of rollback and recovery
- Multiple failures have little effect on the overall calculation
- The number of additional processors required for executing recalculation is estimated: it is not great

Contacts

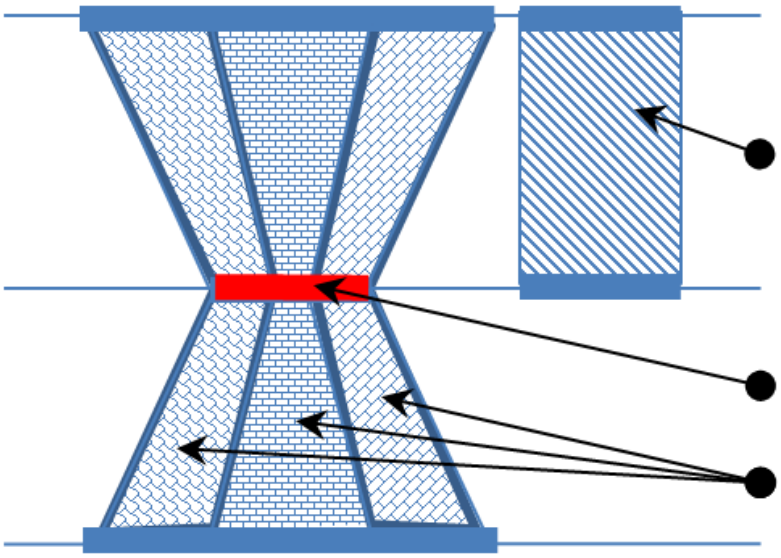
Mikhail Yakobovskiy

Corresponding member of Russian Academy of Sciences.

Deputy director for science in Keldysh Institute of Applied Mathematics,
Russian Academy of Sciences,
Head of department
«Application software for multiprocessor systems and networks»

E-mail: lira@imamod.ru Web: http://lira.imamod.ru

Replace of one failed processor by three reserve processors



Complete recovery of computation process

Continuation of the calculations by operable processors

Processor failure

Lost part of the computational grid

Recalculation of lost data

Local checkpointing record