Исследование масштабируемости параллельного алгоритма численного моделирования удержания плазмы в открытых магнитных ловушках методами имитационного моделирования

Черных Игорь Геннадьевич, Винс Д.В., Вшивков В.А., Боронина М.А. (Институт Вычислительной Математики и Математической Геофизики СО РАН)

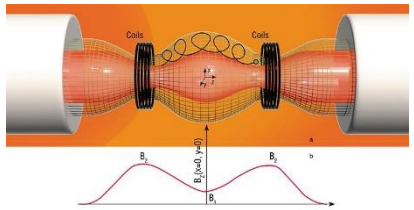
This work was supported by RSF grant No. 19-71-20026

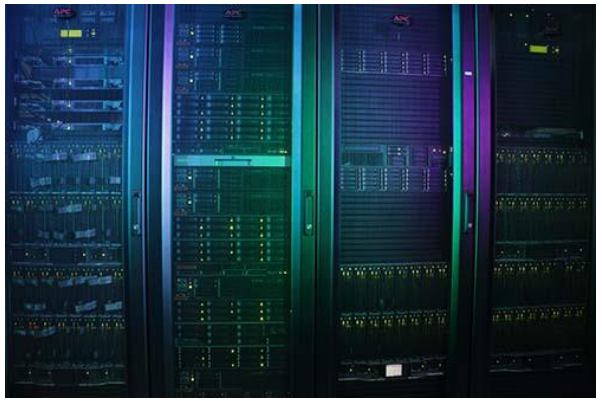
Supercomputer Simulation Laboratory

Igor Chernykh
ICMMG SB RAS

Laboratory plasma

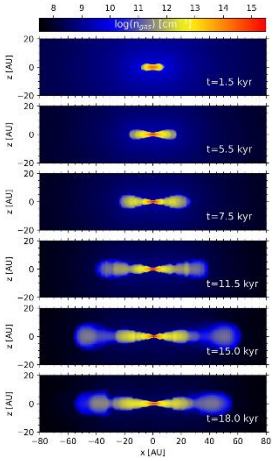
Vitaly Vshivkov
ICMMG SB RAS





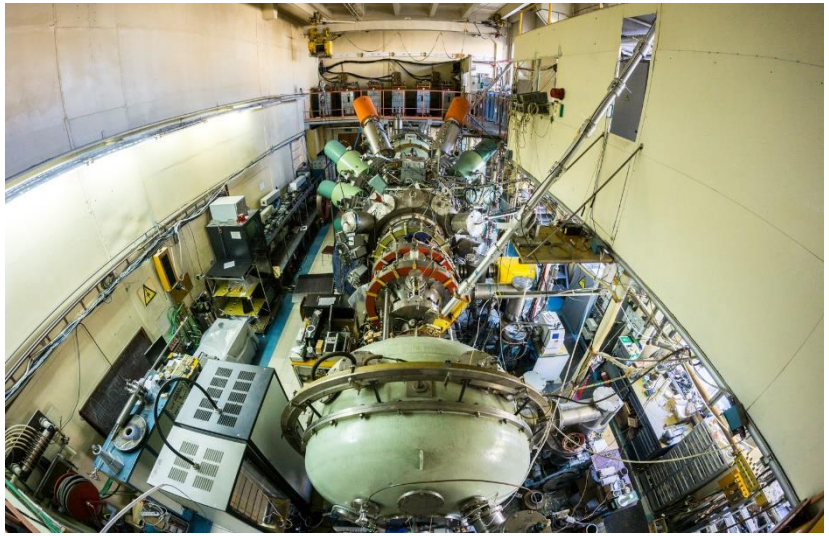
Space Plasma

Igor Kulikov
ICMMG SB RAS

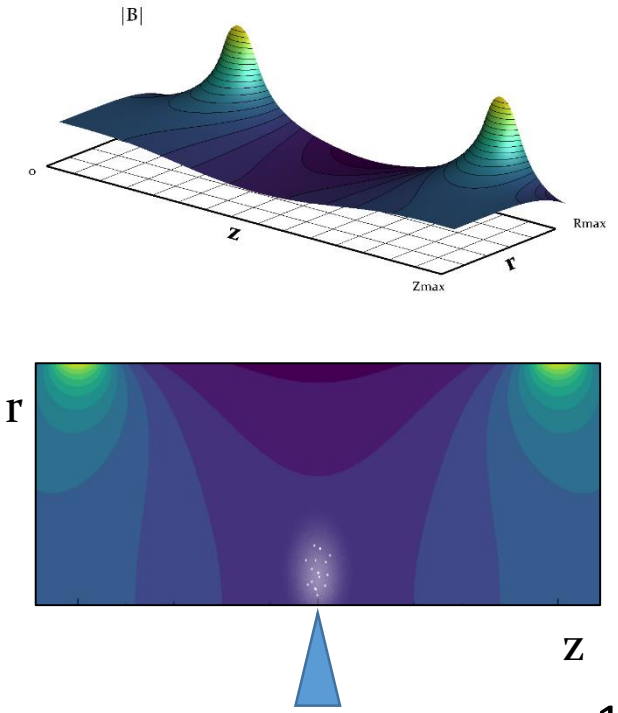


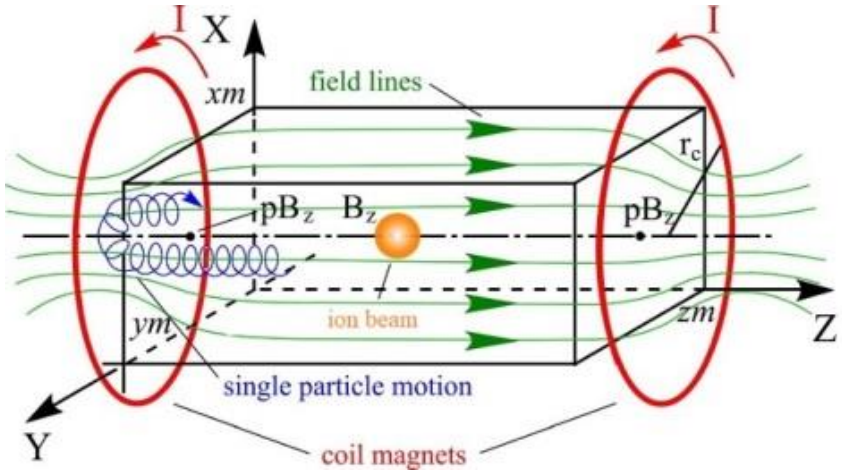
Siberian Supercomputer Center

Overview of the problem



GOL3 – Plasma confinement facility with the external magnetic field generation

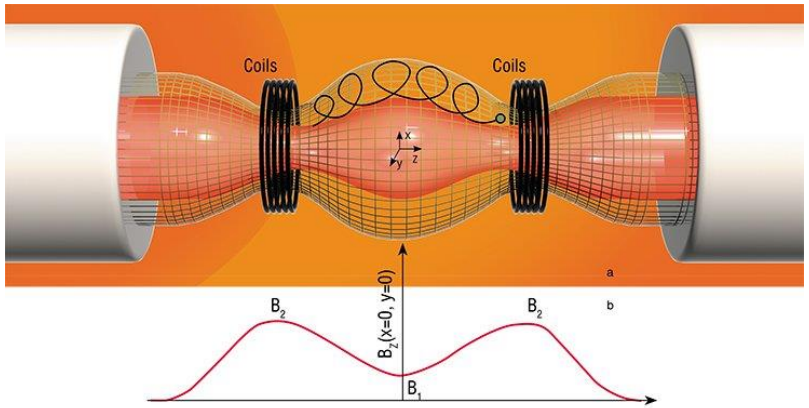




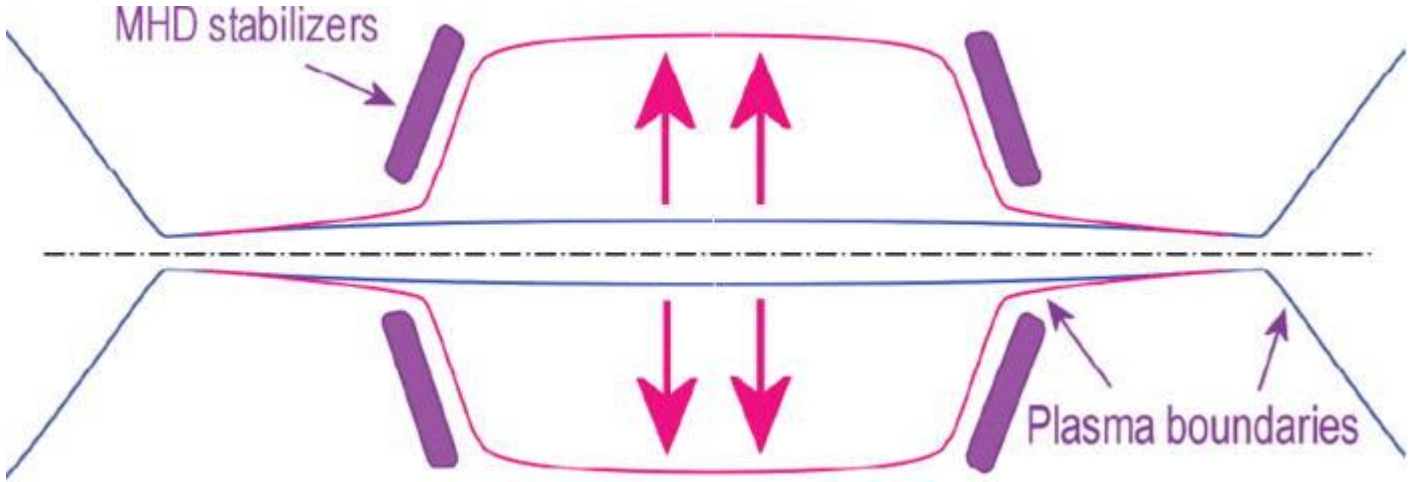
1. Cylindrical volume filled by ions
2. Two magnetic field sources
3. Injection of electrons at the center

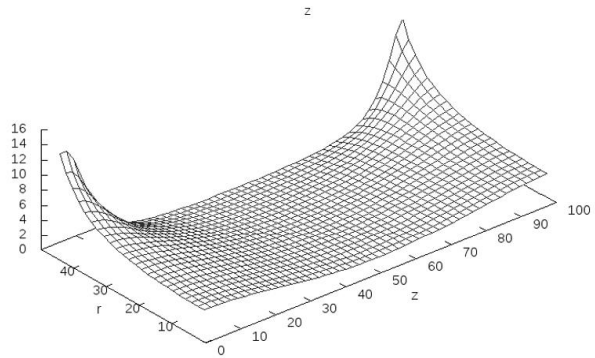
https://en.wikipedia.org/wiki/Budker_Institute_of_Nuclear_Physics

Overview of the problem



A. Shosin 23 Apr 2018 , Budker's Universe , volume Special Issue, N1





beam injection

A. D. Beklemishev **Diamagnetic "bubble" equilibria in linear traps** Phys. Plasmas, 23, 082506 (2016)

Mathematical model

Vlasov kinetics equation (ions)

$$\frac{\partial f_i}{\partial t}+\vec{v}\frac{\partial f_i}{\partial \vec{r}}+\frac{\vec{F}_i}{m_i}\frac{\partial f_i}{\partial \vec{v}}=0$$

$$\vec{F}_i=e\left(\vec{E}+\frac{1}{c}\left[\vec{v},\vec{B}\right]\right)+R_i$$

$$n_i(\vec{r})=\int f_i(t,\vec{r},\vec{v})\,d\vec{v}$$

$$\vec{V}_i(\vec{r})=\frac{1}{n_i(\vec{r})}\int \vec{v} f_i(t,\vec{r},\vec{v})\,d\vec{v}$$

MHD equations (electrons)

$$-e\vec{E}-\frac{e}{c}\left[\vec{V}_e,\vec{B}\right]-\frac{\nabla p_e}{n_e}+\vec{R}_e=m_e\frac{d\vec{V}}{dt}$$

$$\vec{R}_e=-m_e\frac{\vec{V}_e-\vec{V}_i}{\tau_{ei}}$$

Maxwell equations

$$\frac{1}{c}\frac{\partial \vec{E}}{\partial t}=rot\vec{B}-\frac{4\pi}{c}\vec{j}$$

$$\frac{1}{c}\frac{\partial \vec{B}}{\partial t}=-rot\vec{E}$$

$$div\vec{E}=4\pi\rho$$

$$div\vec{B}=0$$

$$n_i=n_e$$

$$R_i=-R_e$$

$$\vec{j}=e\left(n_i\vec{V}_i-n_e\vec{V}_e\right)$$

$$\rho=e\left(n_i-n_e\right)$$

Heat equations

$$n_e\left(\frac{\partial T_e}{\partial t}+\left(\vec{V}_e\nabla\right)T_e\right)=(\gamma-1)\left(Q_e-div\vec{q}_e-p_e div\vec{V}_e\right)$$

$$Q_e=\frac{j^2}{\sigma}$$

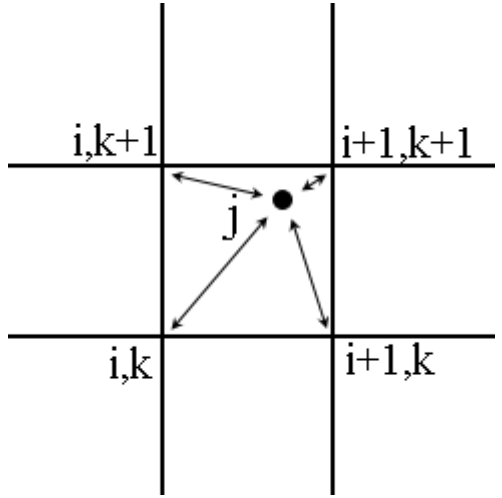
$$\vec{q}_e=-k\nabla Te$$

State equation

$$p_e=n_eT_e$$

Numerical method (Finite-Difference Time-Domain (FDTD)

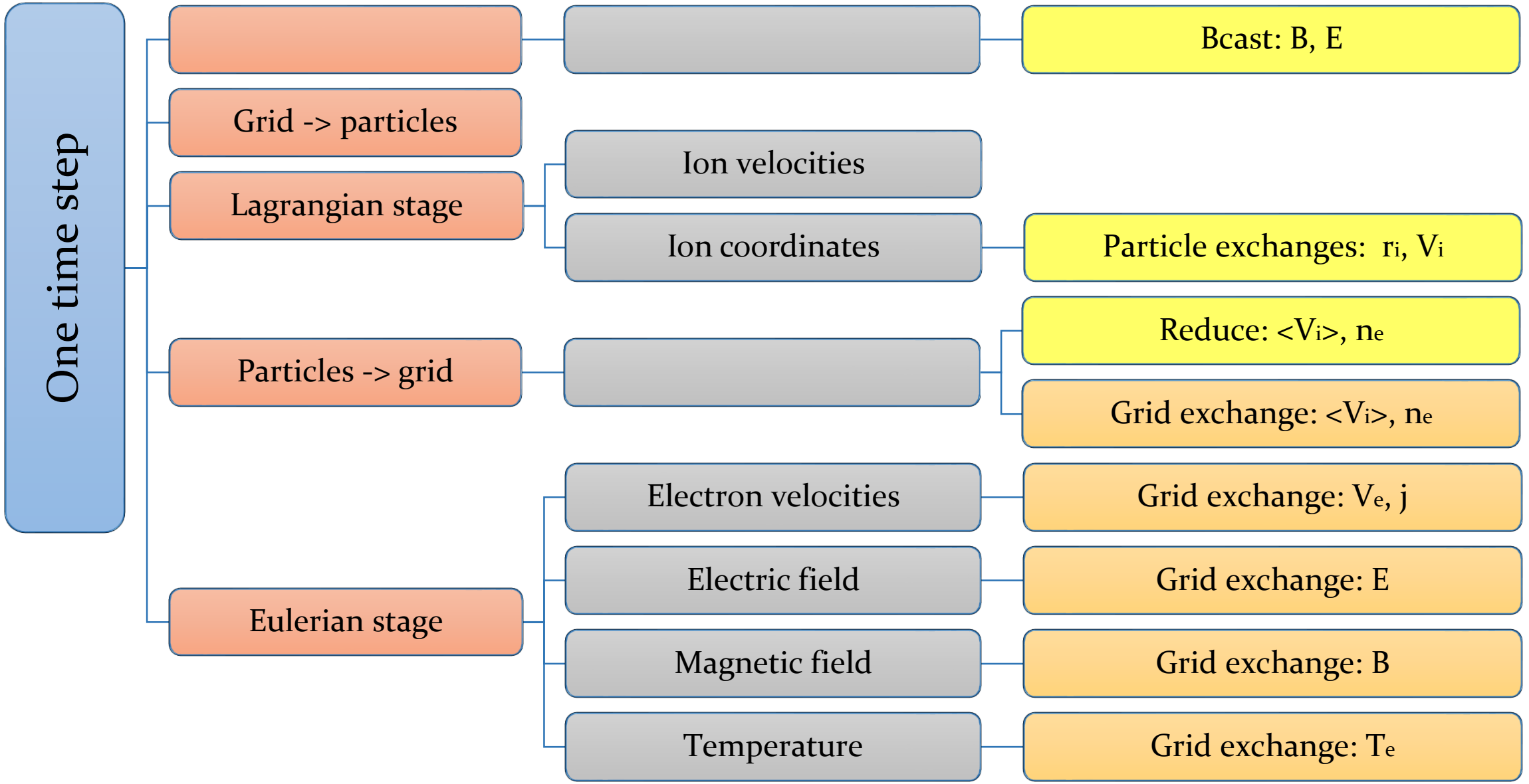
- ✓ Cylinder coordinate system due to the axial symmetry, r=o is singularity
- ✓ Hybrid model, the ions are described kinetically, the electrons by MHD
- ✓ Particles-in-cell method with PIC form-factor
- ✓ Grids shifted by half a step
- ✓Mixed Eulerian-Lagrangian decomposition for the parallelization



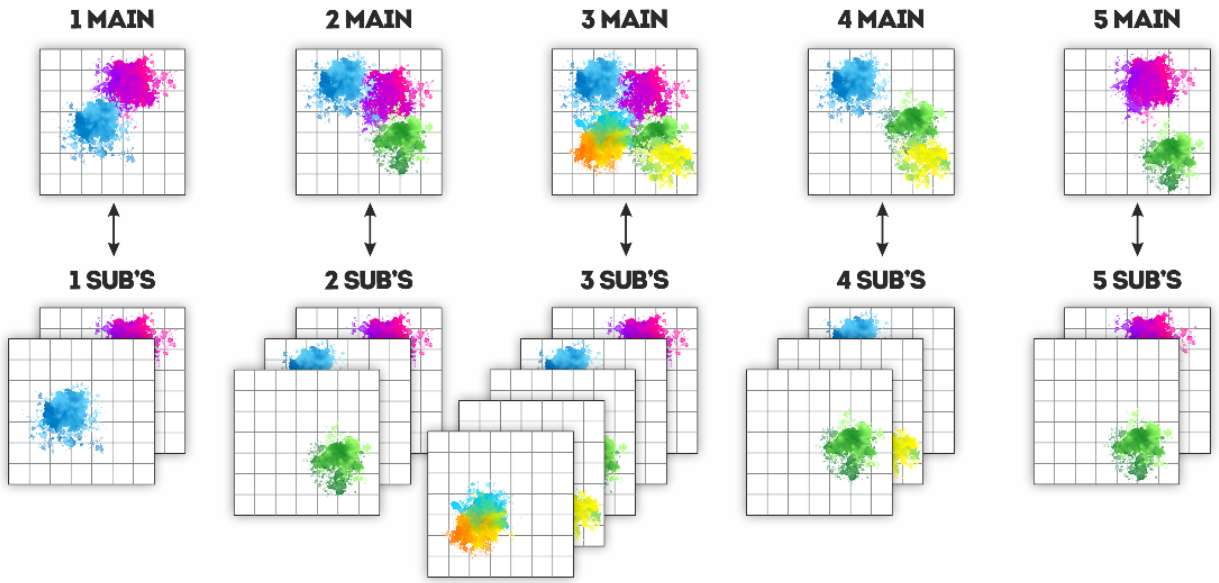
particles >> cells

https://warpx.readthedocs.io/en/latest/theory/picsar_theory.html

Parallel realization



Parallel realization



At the initial moment, the background particles are distributed within the group.

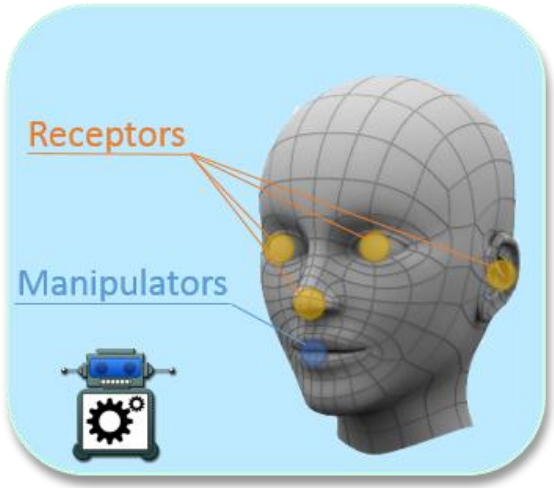
The injected particle is written to the next nucleus in its group.

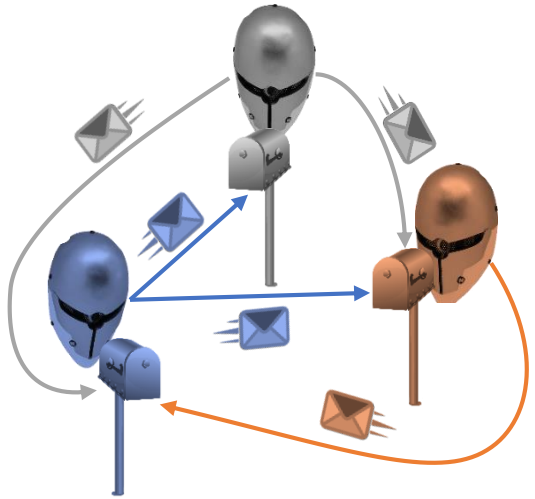
When leaving the subarea, the array of particles is sent to one of the processors of the neighboring group.

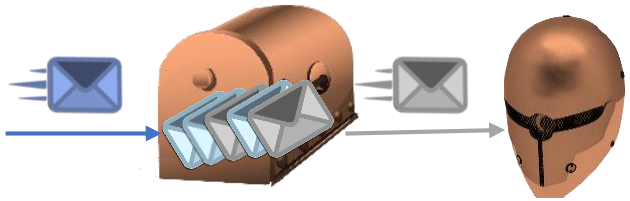
Мультиагентный подход к имитационному моделированию

Агентное моделирование (АМ) – это передовой подход к моделированию систем, содержащих автономных и взаимодействующих интеллектуальных агентов. Интеллектуальный агент – это сущность, живущая в среде обитания, обладающая сенсорами для восприятия среды и исполнительными механизмами для воздействия на среду обитания. Агенты функционируют асинхронно по своим законам, взаимодействуя с другими агентами для достижения общих целей. В процессе функционирования программный агент может изменять как внешнюю среду, так и свое поведение.

Распределённый, асинхронный характер поведения чрезвычайно важен при построении имитационной модели суперкомпьютера, поскольку обеспечить централизованное управление десятками и сотнями миллионов ядер практически невозможно.





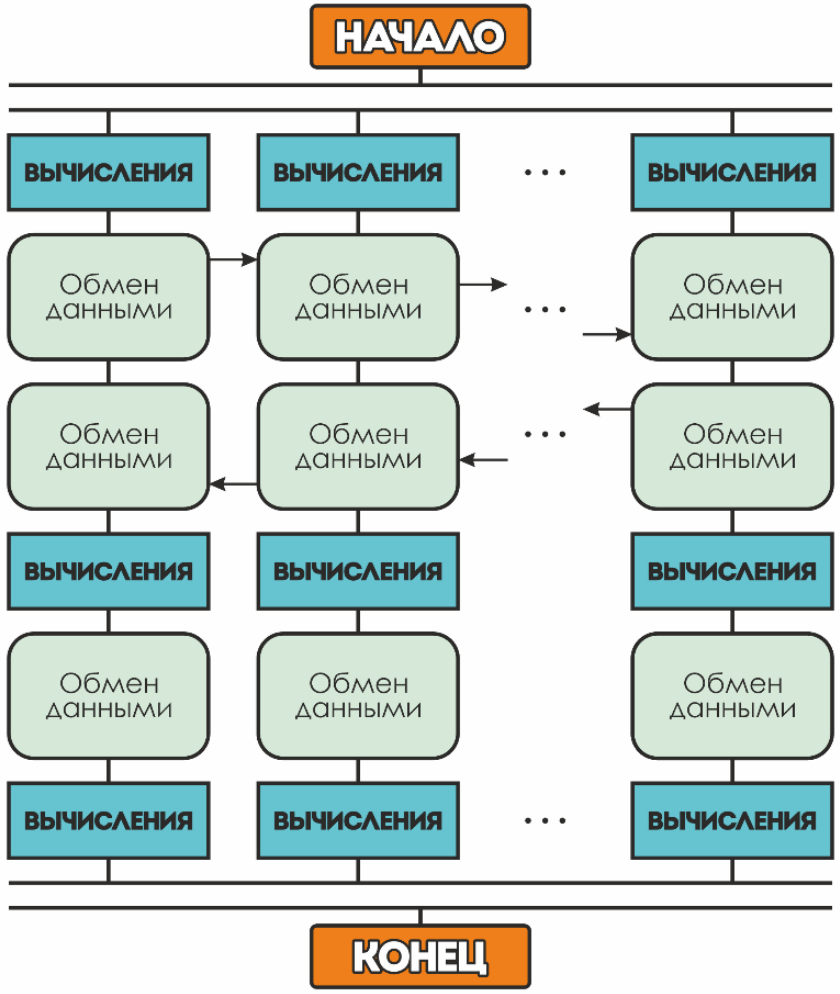


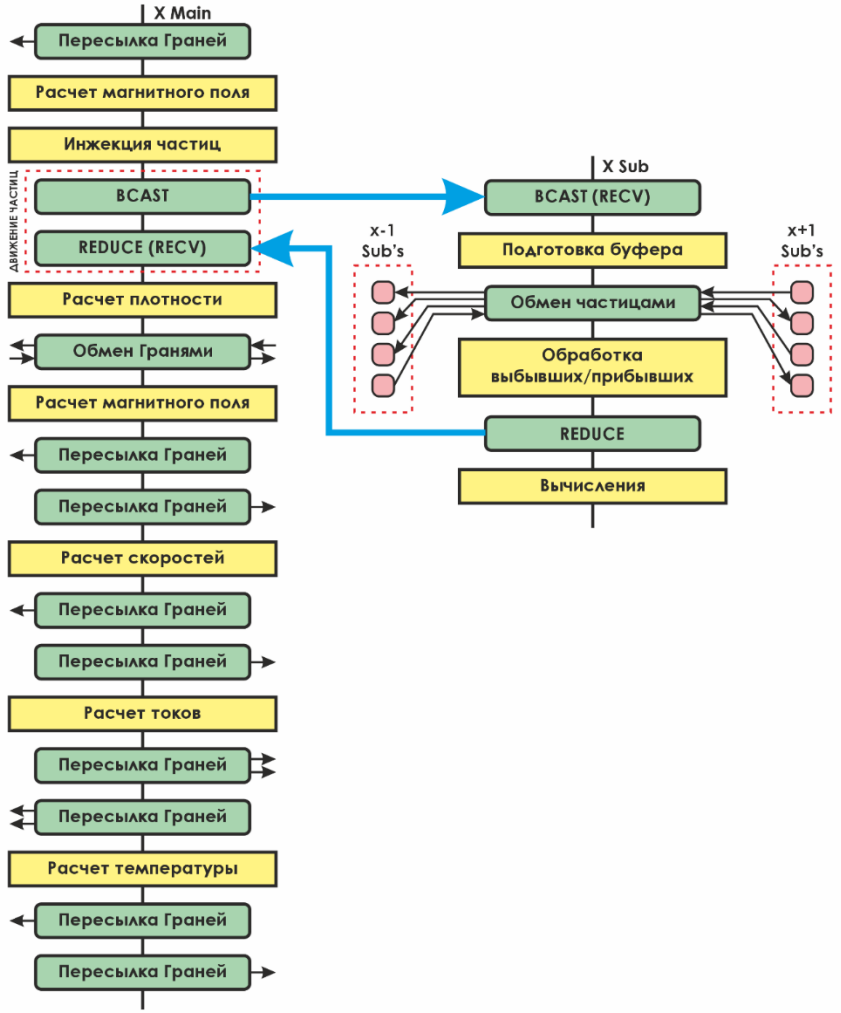
Модель акторов

Модель а́кторов — математическая модель параллельных вычислений, строящаяся вокруг понятия *актора* (англ. *actor* «актёр; действующий субъект»), считающегося универсальным примитивом параллельного исполнения. Актор в данной модели взаимодействует путём обмена сообщениями с другими акторами, и каждый в ответ на получаемые сообщения может принимать локальные решения, создавать новые акторы, посылать свои сообщения, устанавливать, как следует реагировать на последующие сообщения.

По аналогии с философией объектно-ориентированного программирования, где каждый примитив рассматривается как объект, модель акторов выделяет в качестве универсальной сущности понятие «актора». Актор является вычислительной сущностью, которая в ответ на полученное сообщение может одновременно:
отправить конечное число сообщений другим акторам;
создать конечное число новых акторов;
выбрать поведение, которое будет использоваться при обработке следующего полученного сообщения.
Не предполагается существования определённой последовательности вышеописанных действий и все они могут выполняться параллельно.

Модель

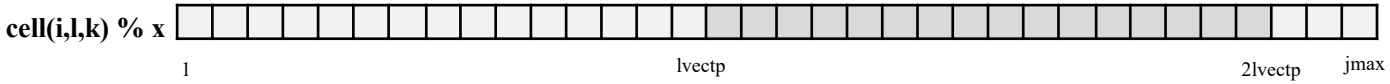




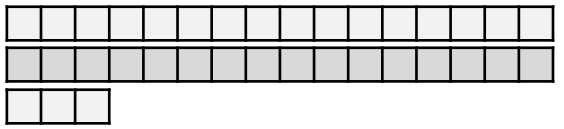
Performance evaluation of 3D code

the particles are bound to the cell cell(i,l,k)
cell(i,l,k) % jmax – number of particles per cell
hole $_{i,l,k}$ % j – the numbers that flew out of the cell
in $_{i,l,k}$ % x(j), y(j)… – data

```
do j=1,   jmax
   dx(j)=…
   dy(j)=…
   dz(j)=…
```



Cache L1
Cascade Lake
data 32kb



do k=1, Kmax
do l=1, Lmax
do i=1, Imax
do j=1, cell(i,l,k)%jmax

- calculation of velocities and coordinates in bundles
- boundary conditions
- incoming / outgoing : Nin, Nout, hole, in
- particle exchanges
- sorting: removing the particles that have flown out and inserting the particles that have flown in, taking into account the injection in the next step.

```
DO jpack=1, jmax, lvectp
 DO jloc=1, MIN(lvectp, jmax - jpack +
1)
     j = jpack+jloc-1
        dx(jloc)=…
        dy(jloc)=…
        dz(jloc)=…
```

Performance evaluation of 3D code

```
257         jEx(jloc) = sx(jloc)*( sy(jloc)*( sz(jloc)*cEx(1,1,1) + dz(jloc)*cEx(1,1,2) )
258     *             +            dy(jloc)*( sz(jloc)*cEx(1,2,1) + dz(jloc)*cEx(1,2,2) ))
259     *           + dx(jloc)*( sy(jloc)*( sz(jloc)*cEx(2,1,1) + dz(jloc)*cEx(2,1,2) )
260     *             +            dy(jloc)*( sz(jloc)*cEx(2,2,1) + dz(jloc)*cEx(2,2,2) ))
```

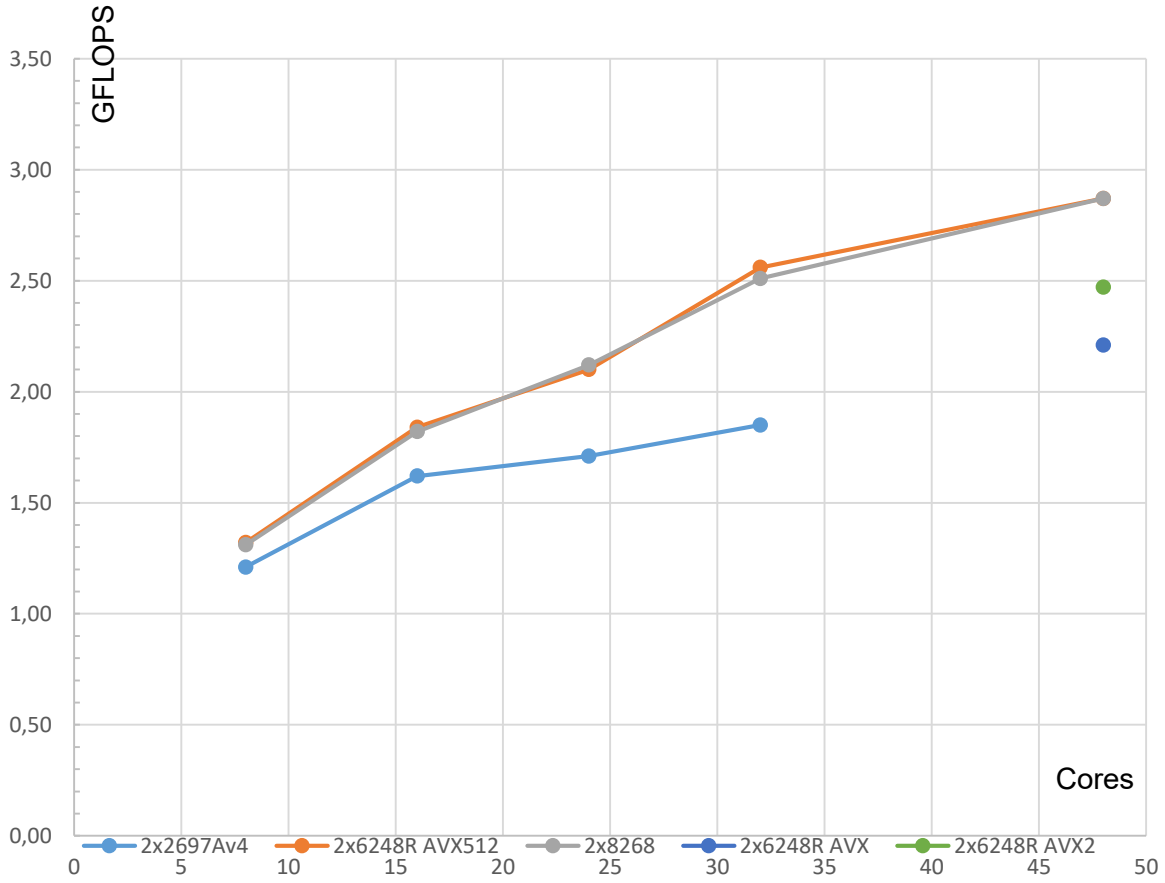
[loop in move_packj at push.f:221]
Vectorized (Body) AVX512; processes Float32; Float64 data type(s)
Performance: **10.951** GFLOPS
Self Time: **12.296** s
Self Elapsed Time: **12.296** s
Total Time: **12.296** s
Total Elapsed Time: **12.296** s
Self Memory Traffic: **396.151** GB
L1 Arithmetic Intensity: **0.34** FLOP/Byte

Module: cloud3dm.out!0x462598

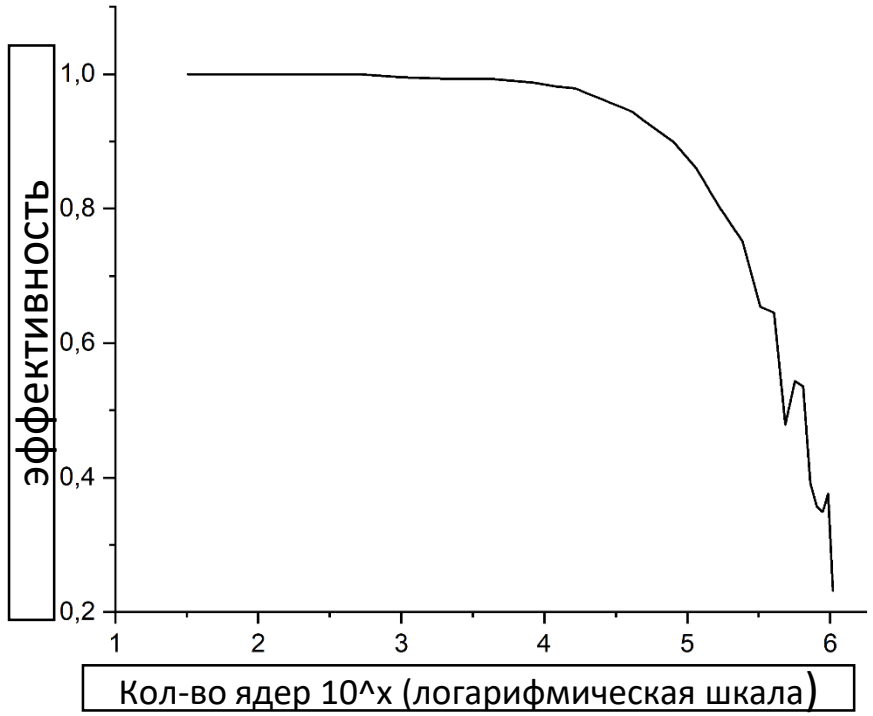
Address	Lin.	
0x462755	254	vfmadd213pd %zmm7, %zmm8, %k0, %zmm31
0x46275b	260	vmovupsz -0x7f0(%rbp), %k0, %zmm7
0x462765	257	vfmadd231pd %zmm10, %zmm19, %k0, %zmm9
0x46276b	252	vfmadd213pd %zmm3, %zmm8, %k0, %zmm30

[loop in move_packj at push.f:221]
Vectorized (Body) AVX512; processes Float32; Float64 data type(s)
Performance: **12.19** GFLOPS
Self Time: **11.046** s
Self Elapsed Time: **11.046** s
Total Time: **11.046** s
Total Elapsed Time: **11.046** s
Self Memory Traffic: **396.151** GB
L1 Arithmetic Intensity: **0.34** FLOP/Byte

Performance evaluation of 3D code



Результаты имитационного моделирования

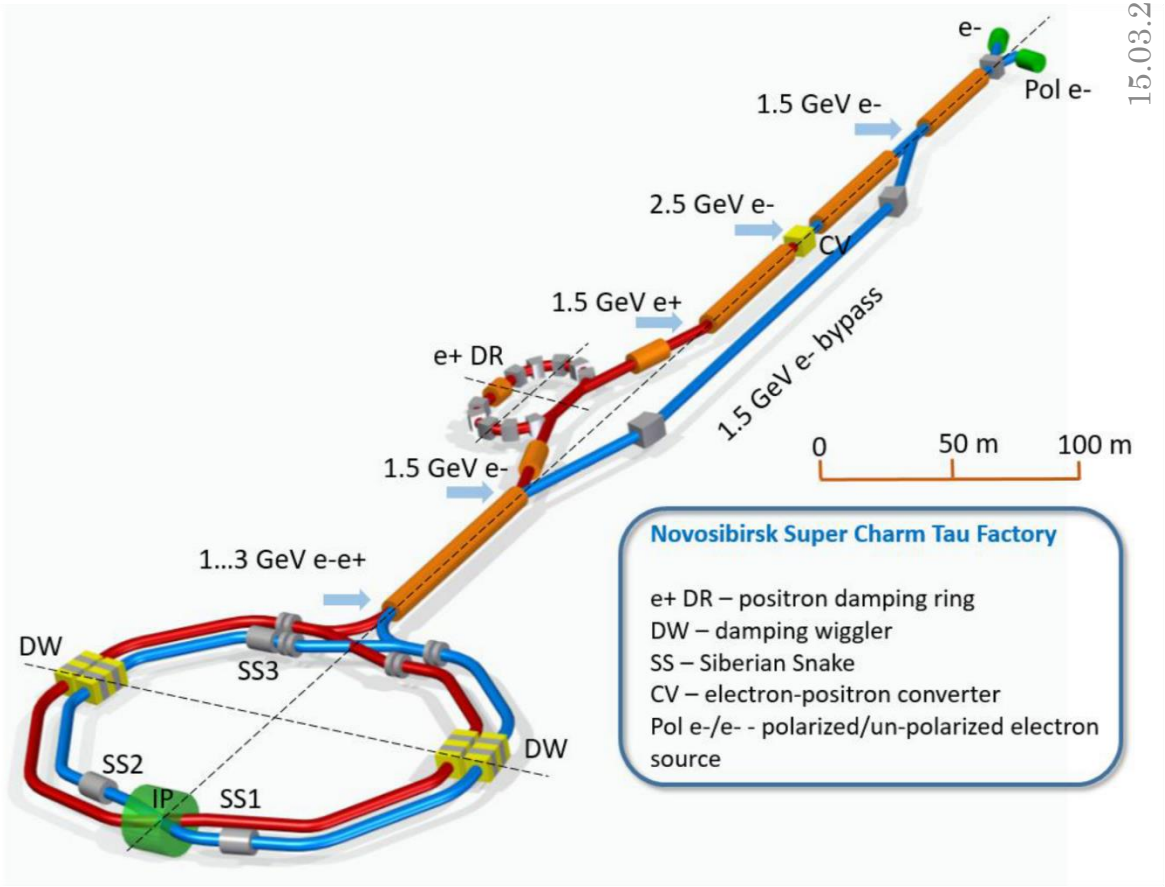


Проект «Супер Чарм-Тау Фабрика» (СЧТФ).

В ИЯФ СО РАН разрабатывается проект «Супер С-тау фабрики» – симметричного электрон-позитронного коллайдера ультравысокой светимости с энергией пучков в системе центра масс от 2 до 6 ГэВ.

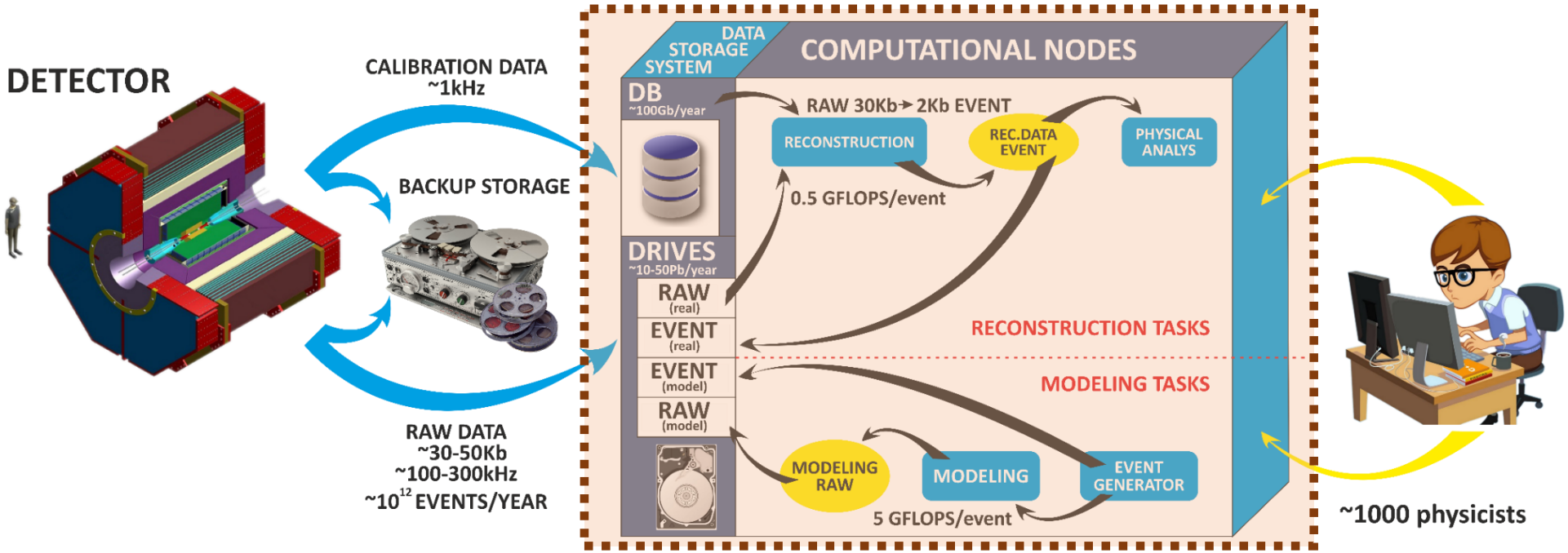
Проект включает в себя уникальный ускорительно-накопительный комплекс со светимостью 10^{35} $см^{-2}с^{-1}$ и универсальный детектор элементарных частиц.

Основная задача экспериментов на Супер С-тау фабрике – изучение свойств тау-лептона, «очарованных» частиц, прецизионная проверка существующей теории микромира, Стандартной Модели и поиск феноменов, не описываемых в рамках этой теории.

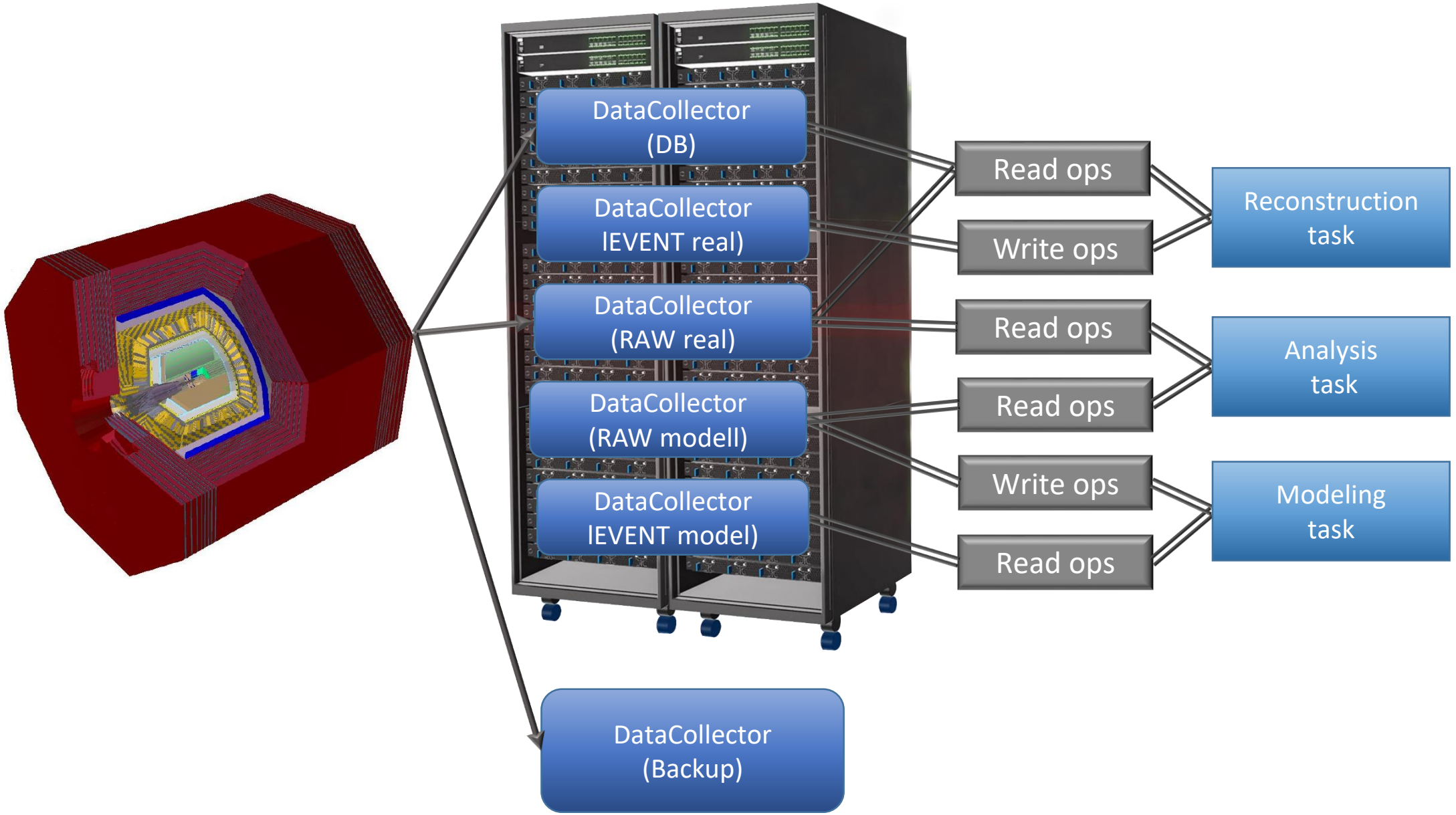


СЧТФ. Требования к вычислительной инфраструктуре (ВИ).

В ходе проведения экспериментов с детектора элементарных частиц СЧТФ будет считано порядка 100 петабайт «RAW» данных. Важную роль в проекте играет система обработки и хранения данных, в задачи которой входит первичная обработка данных, передача данных в систему долговременного хранения данных (десятки лет), извлечение данных из системы хранения для обработки и обработка с использованием HPC. Специализированное программное обеспечение должно позволить проводить анализ набранных данных коллективом порядка 1000 физиков.

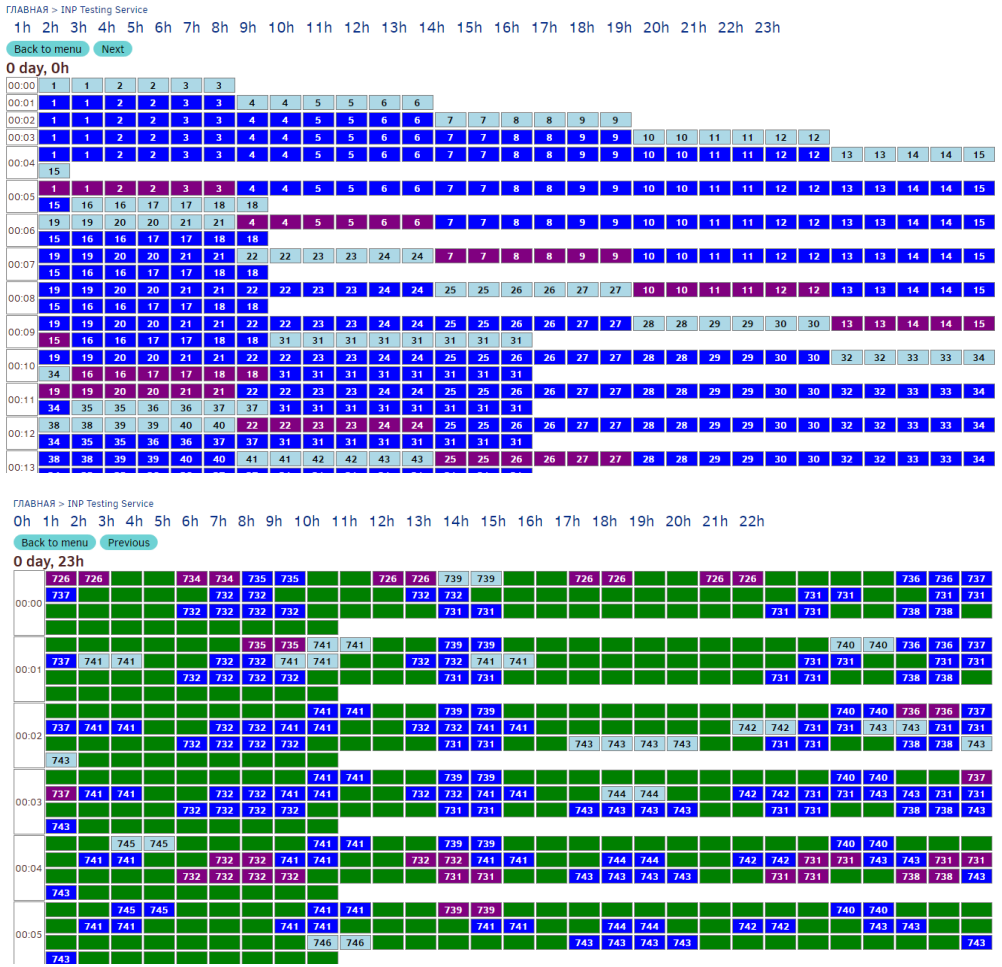


Модель ВИ СЧТФ. Система хранения данных.



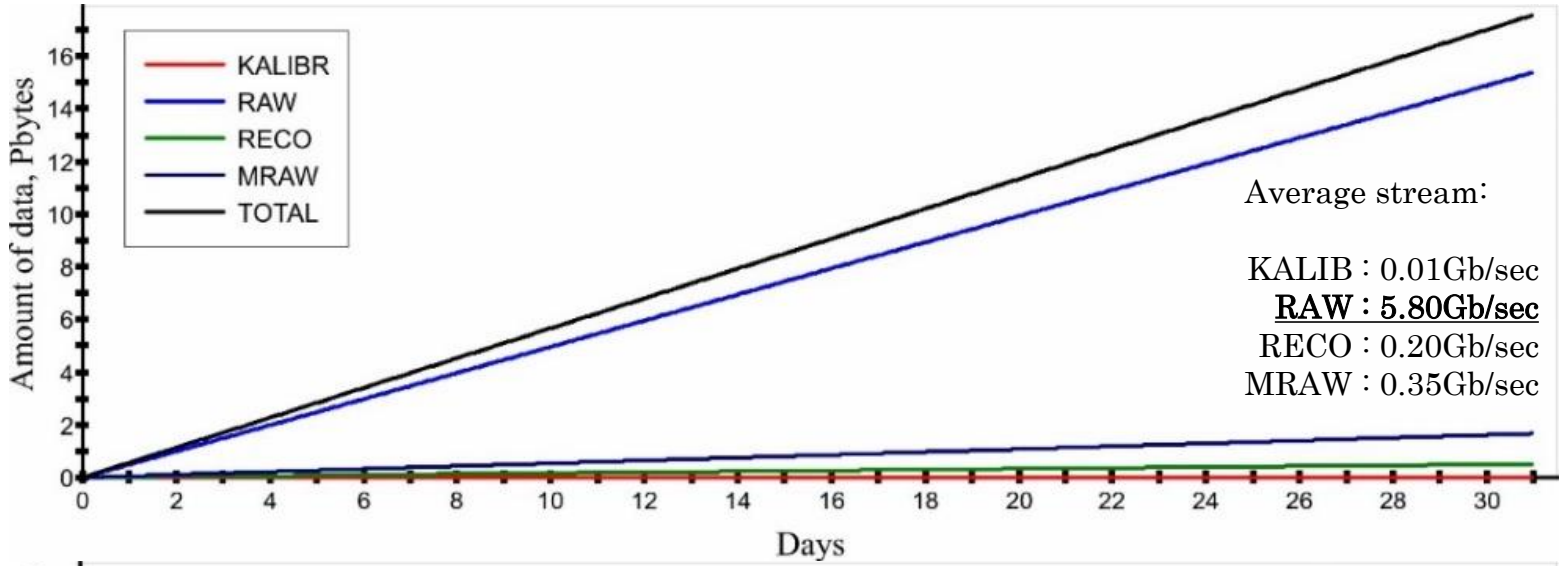
Модель ВИ СЧТФ. Агент анализа статистики.

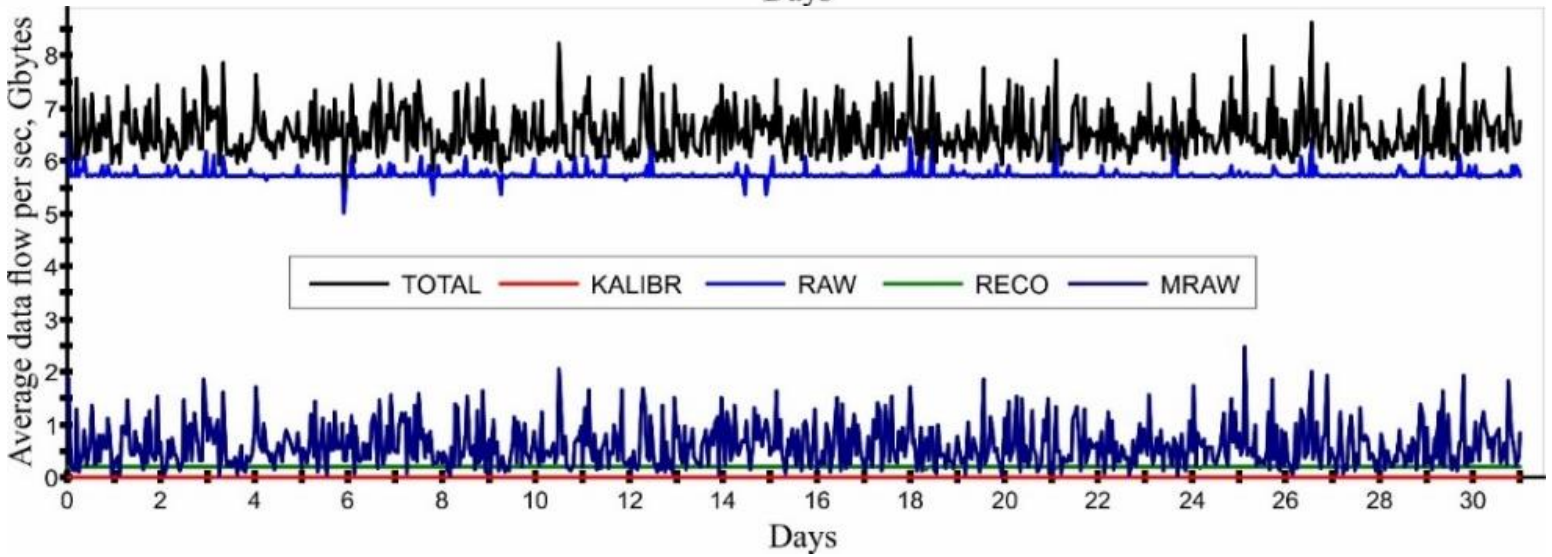
На основе SQL+JS+PHP создан модуль для интерактивного просмотра статистики, снимаемой агентом анализа статистики. Просмотр состояния узлов (/сек):



Анализ полного объема данных и среднего потока данных

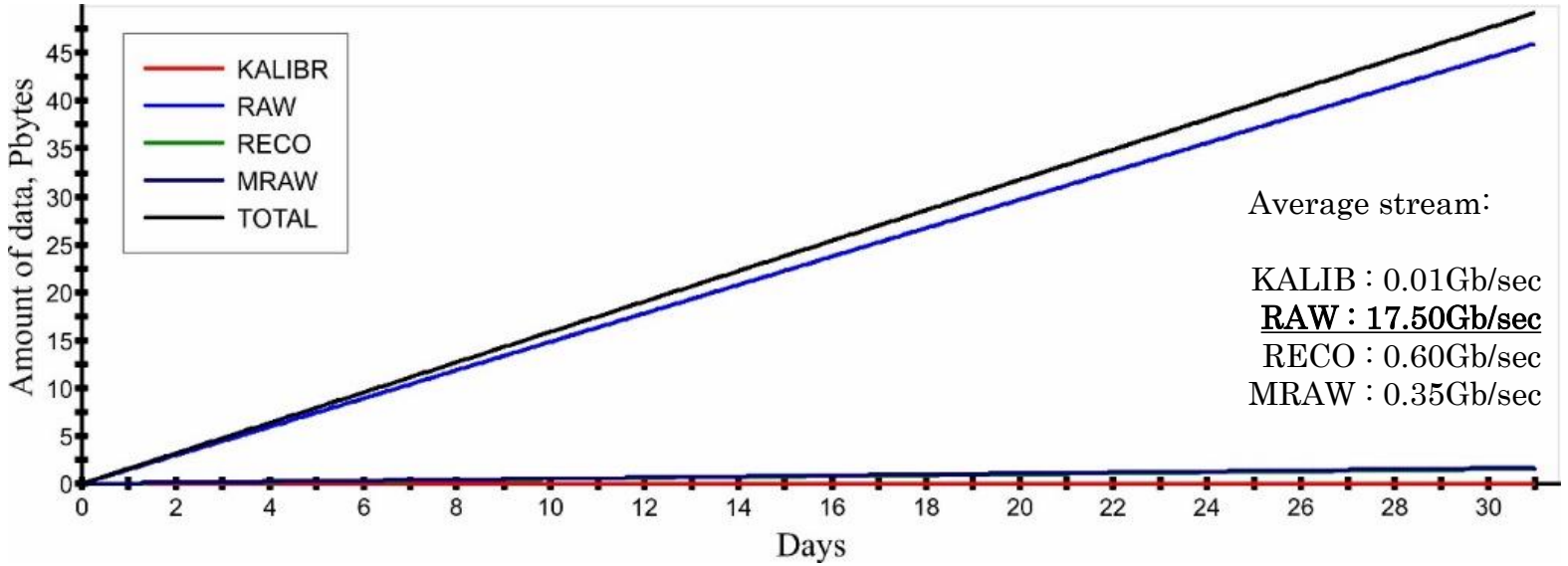
F_{data} = 100kHz; ~5Pbytes/10days; ~17.5Pbytes/31days

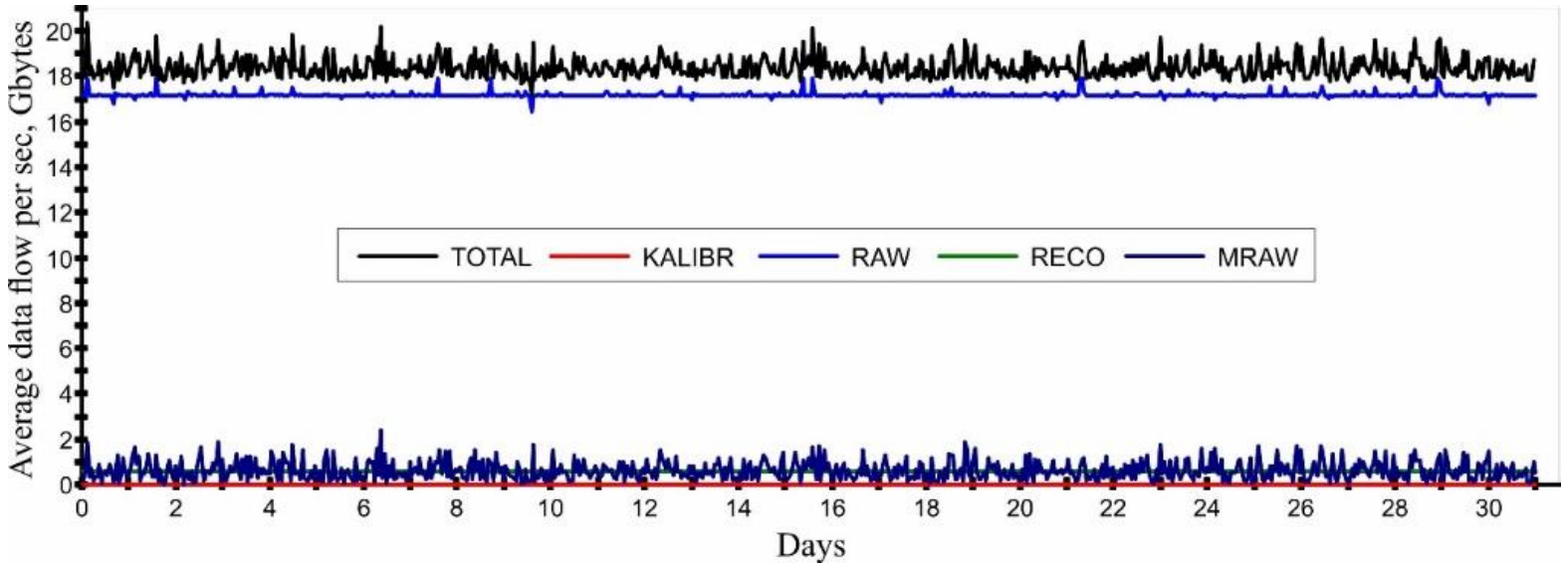




Анализ полного объема данных и среднего потока данных

F_{data} = 300kHz; ~16Pbytes/10days; ~50Pbytes/31days;

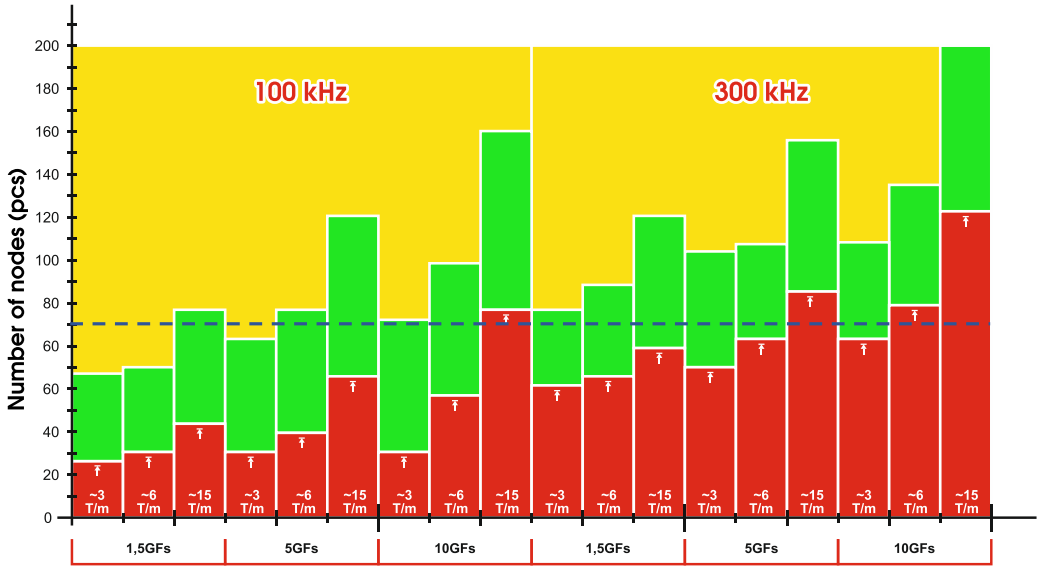


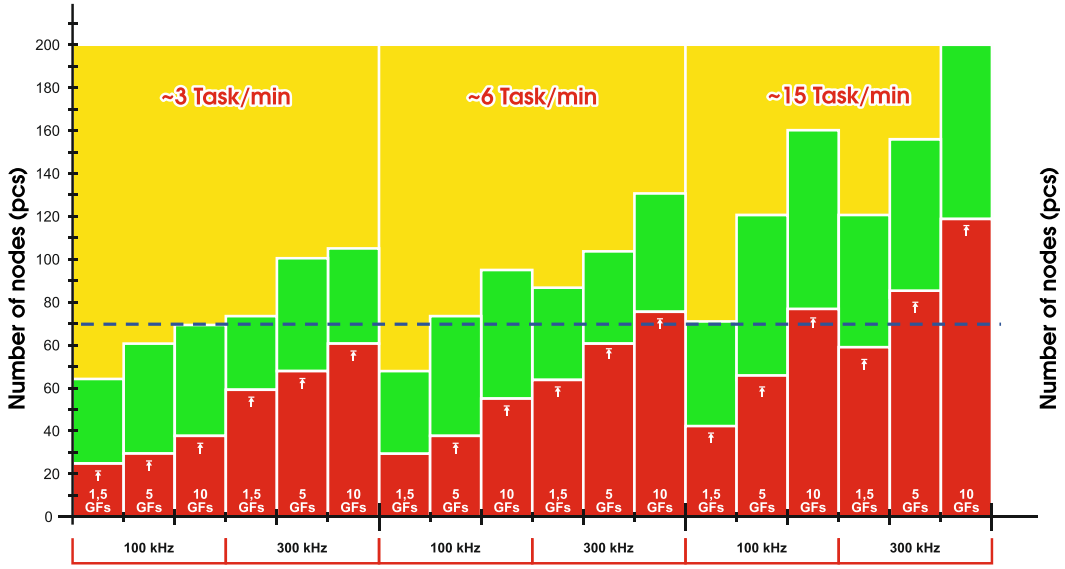


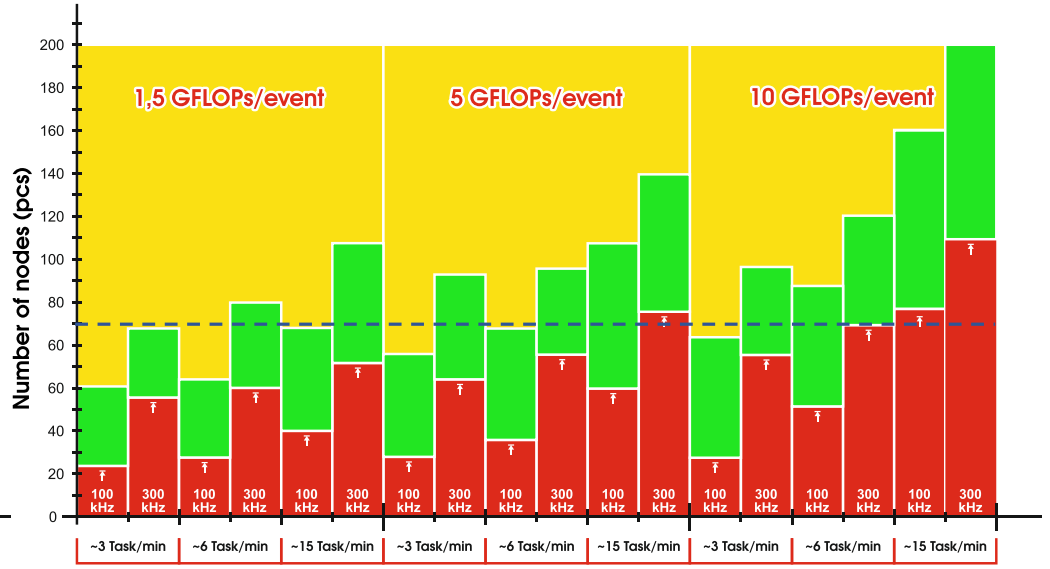
Анализ оптимального числа вычислительных узлов

Для различных режимов и характеристик потока задач 3 диапазона:

- Красный – нехватка свободных узлов, очередь перегружена;
- Желтый – избыток свободных узлов, неэффективное использование;
- Зеленый – оптимальное число узлов.



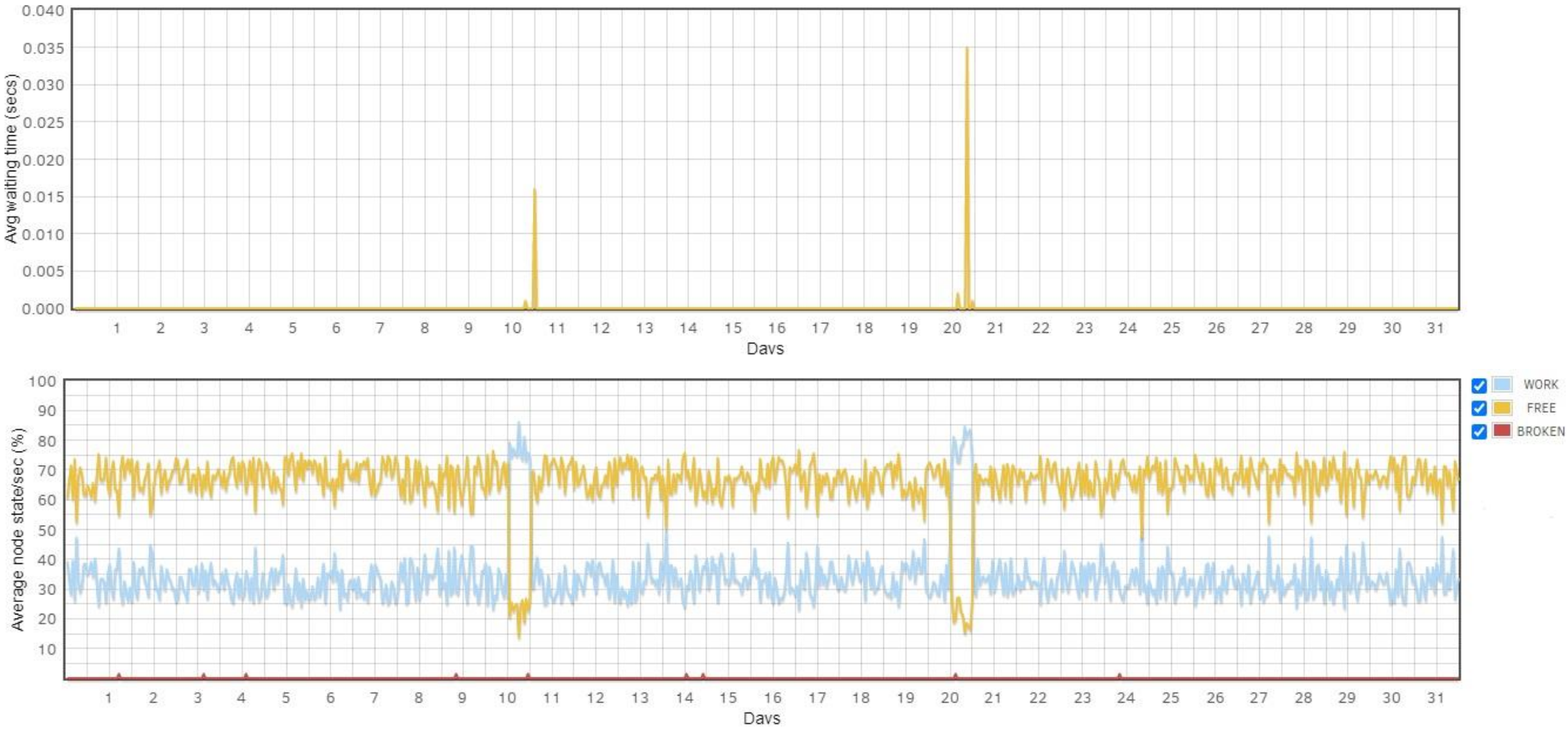




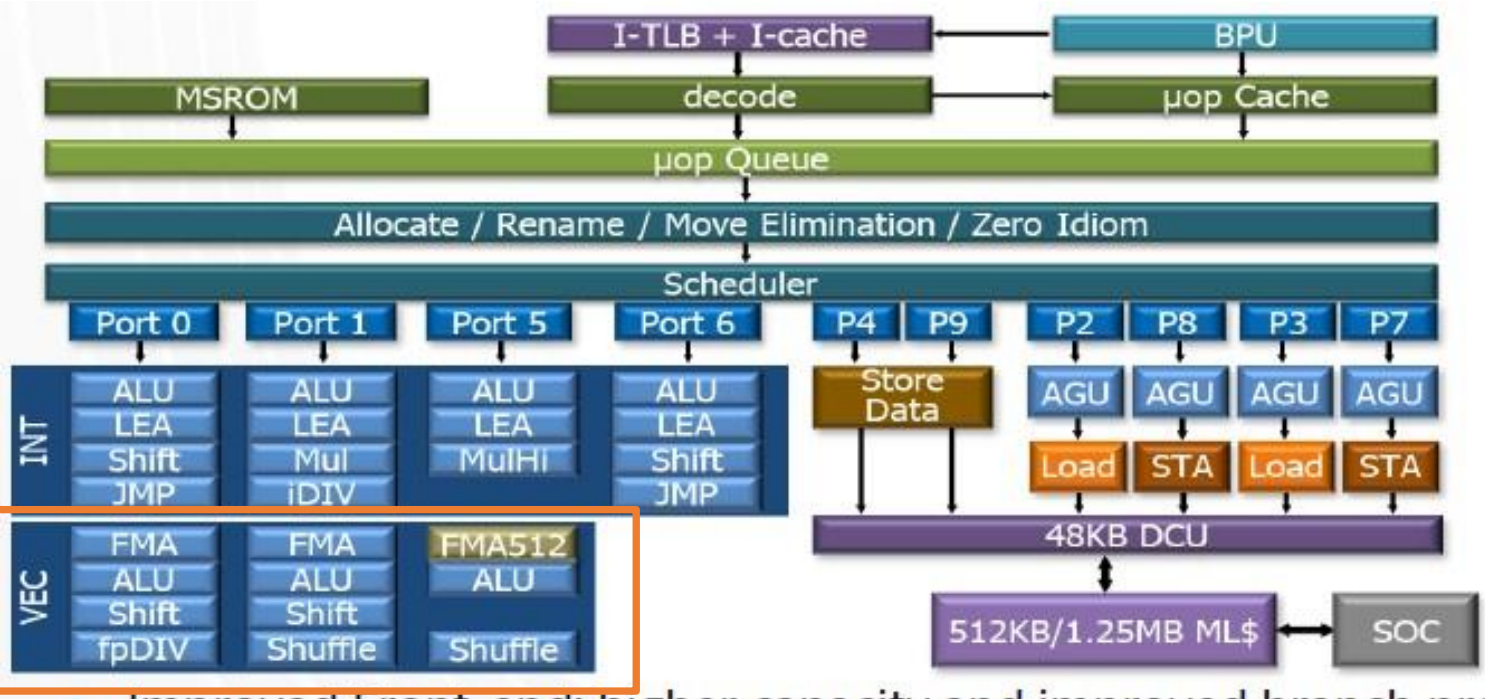
Полномасштабное моделирование инфраструктуры

выбранные параметры и гибридный режим работы

N_{nodes} = 70; F_{data} = 100/300kHz; C_{model}=2.5GFLOPs/event; P_{model} =10%(~6Tasks/min);



Thank you for your attention

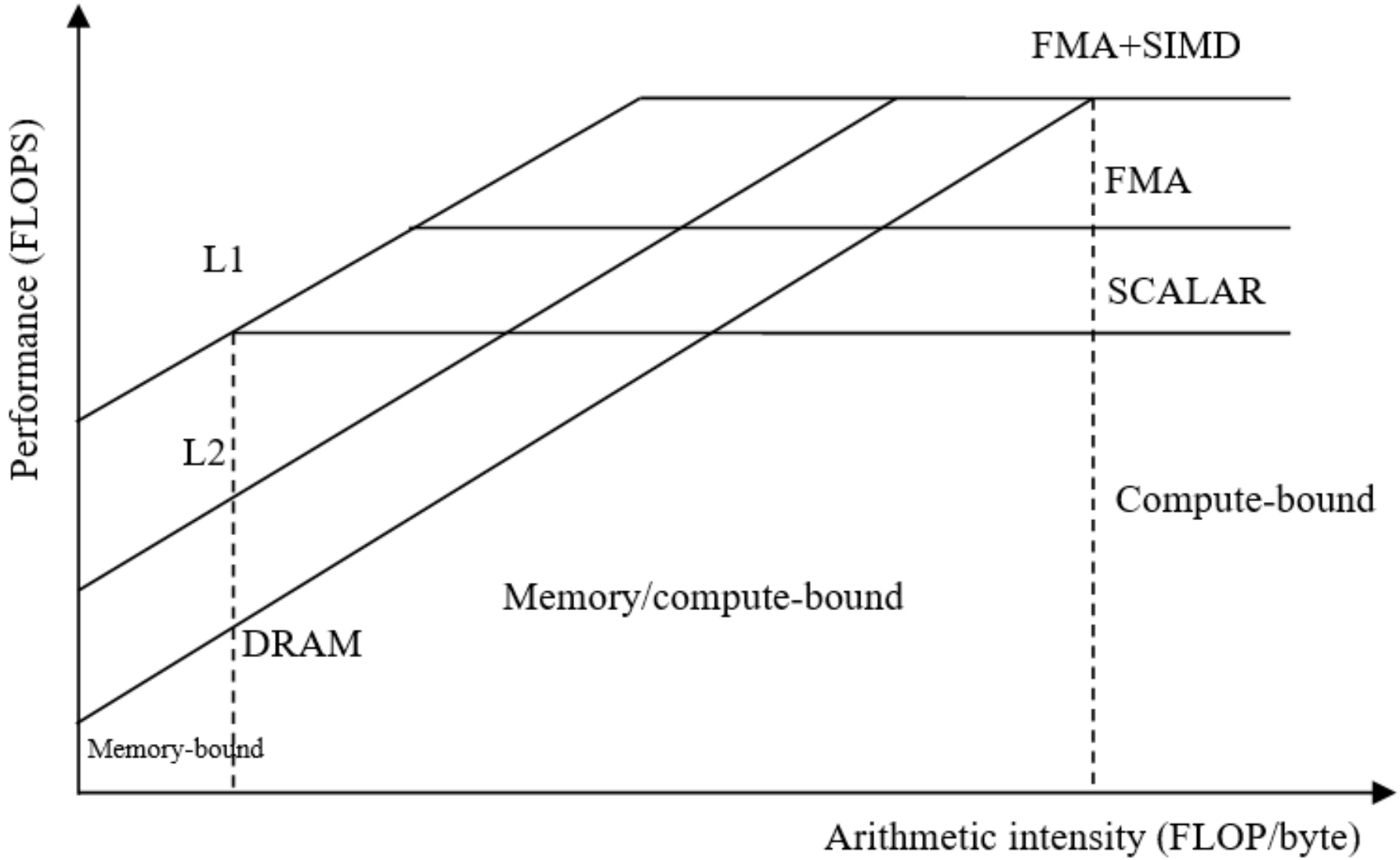


	Cascade Lake (per core)	Ice Lal (per core)
Out-of-order Window	224	384
In-flight Loads + Stores	72 + 56	128 + 7
Scheduler Entries	97	160
Register Files – Integer + FP	180 + 168	280 +2
Allocation Queue	64/thread	70/thre 140/1 thr
L1D Cache (KB)	32	48
L1D BW (B/Cyc) – Load + Store	128 + 64	128 + 6
L2 Unified TLB	1.5K	2K
Mid-level Cache (MB)	1	1.25

- Improved Front-end: higher capacity and improved branch predictor
- Wider and deeper machine: wider allocation and execution resources + larger structures
- Enhancements in TLBs, single thread execution, prefetching
- Server enhancements – larger Mid-level Cache (L2) + second FMA

~18% Increase In IPC On Existing SPECcpu2017(est) Integer Rate Binaries

Roofline model



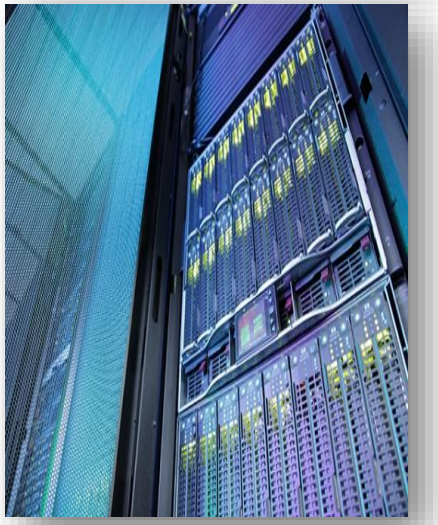
Siberian Supercomputer Center

НКС-1П (РСК, горячая вода, 2448 ядер, ~182ТФЛОПС Rpeak):

- 27 узлов: 2 CPU Intel Xeon E5-2697v4 [128 GB DDR4, 256 GB DDR4] (864 ядер, 2.6GHz) (1 узел 2x375GB Intel Optane [IMDT]) (с учетом узлов ИГиЛ СО РАН)
- 16 узлов: 1 CPU Intel Xeon Phi 7290 KNL [16 GB MCDRAM+96 GB DDR4] (1152 ядер, 1.5-1.7 GHz)
- 1 узел: 2 CPU Intel Xeon Platinum 8268 [192 GB DDR4] (48 ядер, 2.9 GHz)
- 10 узлов: 2 CPU Intel Xeon Gold 6248R [192/384/768 GB DDR4] (480 ядер, 2.9 GHz) (с учетом узлов лабораторий, ИГиЛ СО РАН)
- Intel OmniPath 100 Gb/s
- Intel Lustre – 200 TB + NFS 100TB(ИГиЛ СО РАН)

НКС-30Т (HP, воздушное охлаждение, ~1500 CPU ядер (2.9GHz), ~61000 GPU ядер, 85ТФЛОПС (сегмент с GPU) + 22ТФЛОПС (сегмент CPU)):

- 576 CPU Intel Xeon E5450/E5540(2688 ядер)
- 80 CPU Intel Xeon X5670(480 ядер)
- 120 GPU NVIDIA Tesla M 2090(61440 ядер)
- Infiniband QDR 40 Gb/s
- HP Ibrix – 90 TB



Performance evaluation

Problems with code: else if statements, difficult data balancing, arrays

Structure of arrays and data alignment:

```
type coo
real*8, allocatable :: r(:)
real*8, allocatable :: z(:)
real*8, allocatable :: u(:)
real*8, allocatable :: v(:)
real*8, allocatable :: w(:)
real*8, allocatable :: a(:)
real*8, allocatable :: it(:)
dir$ attributes align:64 :: r
dir$ attributes align:64 :: z
dir$ attributes align:64 :: u
dir$ attributes align:64 :: v
dir$ attributes align:64 :: w
dir$ attributes align:64 :: a
end type coo
```

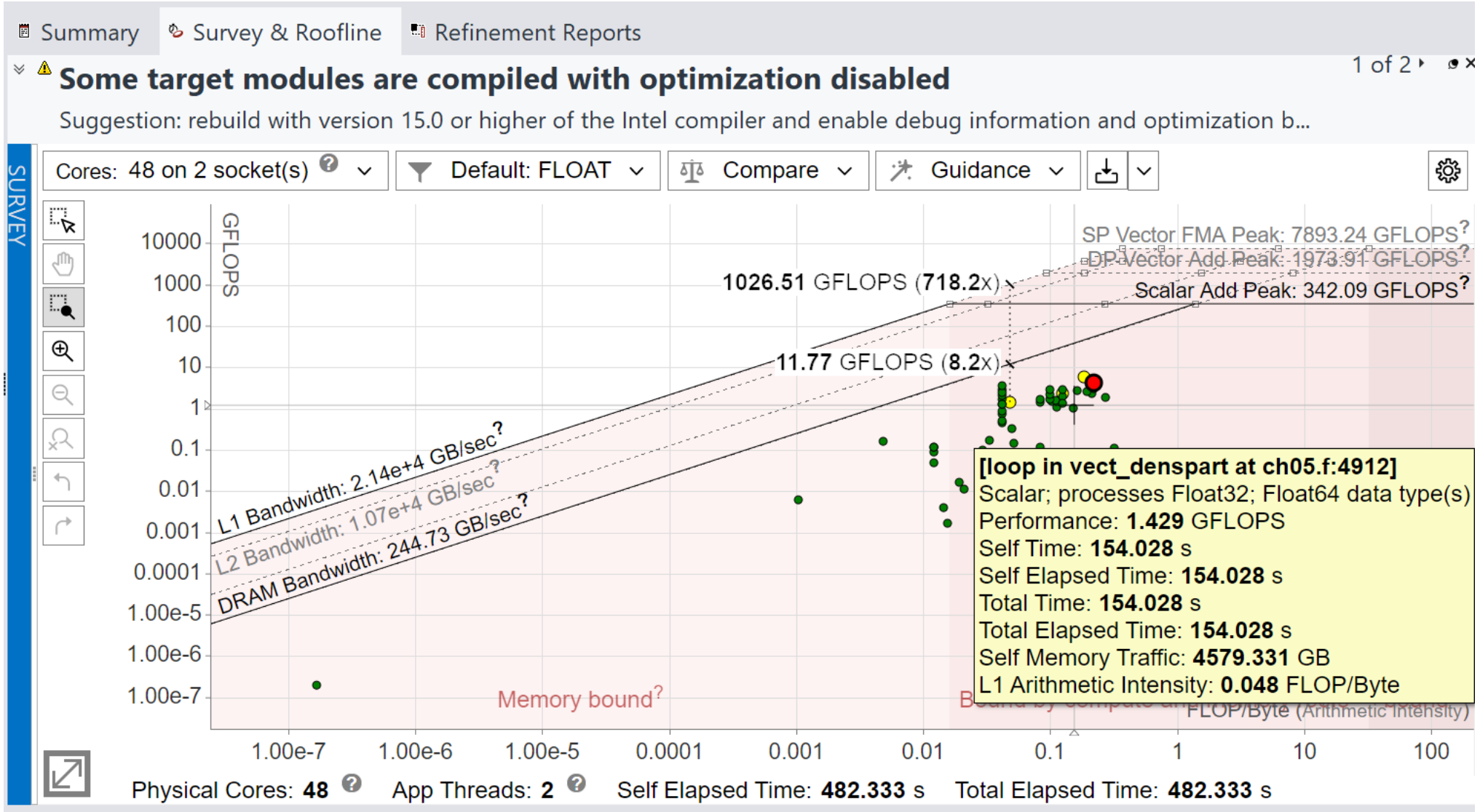
Three versions of code:

1. No aligned data , no simd instructions, Intel Fortran compiler 2019 mpiifort

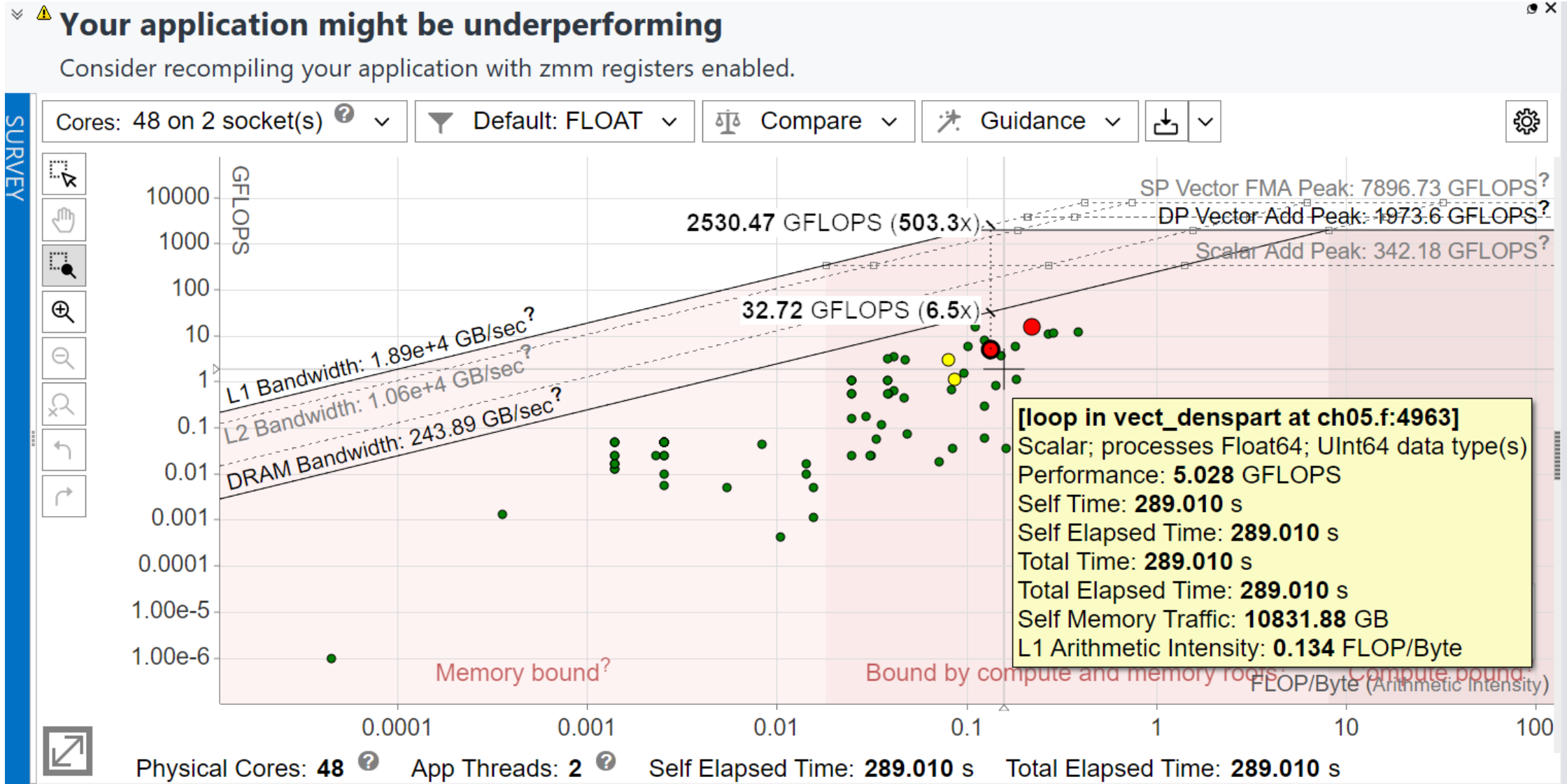
2. Aligned data, AVX512, Intel Fortran compiler 2019 mpiifort -xcascadelake

3. No aligned data, AVX512, Intel Fortran oneAPI 2021 compiler mpiifort -xcascadelake

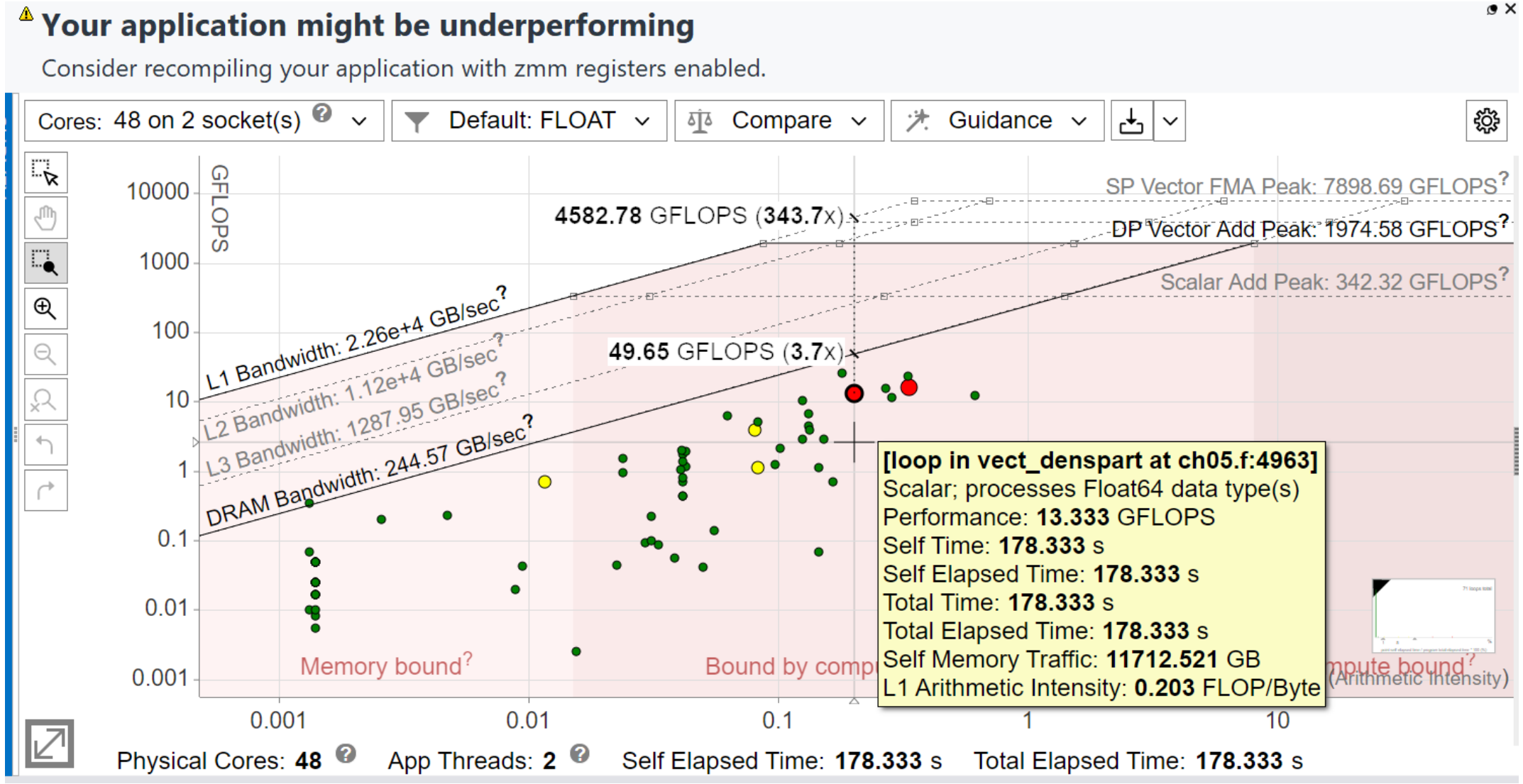
Roofline model of parallel code (2x6248R, nonaligned, nonsimd, Intel Fortran compiler 2019 (1.23 GFLOPS)



Roofline model of parallel code (2x6248R, aligned, AVX-512, Intel Fortran compiler 2019) 1.86 GFLOPS



Roofline model of parallel code (2x6248R, nonaligned, AVX-512, Intel Fortran oneAPI 2021) 2.66 GFLOPS



Performance evaluation

Particles density calculation function (Eulerian stage):

1. vec_denspart (nonaligned, nonsimd, Intel Fortran compiler 2019) – 1.43 GFLOPS
2. vec_denspart(aligned, AVX512, Intel Fortran compiler 2019) – 5 GFLOPS (3.5x)
3. vec_denspart (nonaligned, AVX512, Intel Fortran oneAPI 2021) – 13.33 GFLOPS (2.66x)

Structure of arrays, data alignment, compiler autovectorization

Total code performance:

1. PIC code (nonaligned, nonsimd, Intel Fortran compiler 2019) – 1.23 GFLOPS
2. PIC (aligned, AVX512, Intel Fortran compiler 2019) – 1.86 GFLOPS (+51%)
3. PIC (nonaligned, AVX512, Intel Fortran oneAPI 2021) – 2.66 GFLOPS (+43%)

Conclusion

1. **Update your system software and compilers.**
2. **Use advanced vector instructions (AVX2, AVX512, AMX) together with MPI and OpenMP in your code.**
3. **Use all compiler optimization options to build the fastest code for your CPU architecture. (mpiifort –xCOMMON-AVX512)**
4. **Use FMA commands**

COMMON-AVX512	May generate Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Foundation instructions, Intel® AVX-512 Conflict Detection Instructions (CDI), as well as the instructions enabled with CORE-AVX2. Optimizes for Intel® processors that support Intel® AVX-512 instructions.
CORE-AVX512	May generate Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Foundation instructions, Intel® AVX-512 Conflict Detection Instructions (CDI), Intel® AVX-512 Doubleword and Quadword Instructions (DQI), Intel® AVX-512 Byte and Word Instructions (BWI) and Intel® AVX-512 Vector Length Extensions (VLE), as well as the instructions enabled with CORE-AVX2. Optimizes for Intel® processors that support Intel® AVX-512 instructions.
CORE-AVX2	May generate Intel® Advanced Vector Extensions 2 (Intel® AVX2), Intel® AVX, SSE4.2, SSE4.1, SSE3, SSE2, SSE, and SSSE3 instructions for Intel® processors. Optimizes for Intel® processors that support Intel® AVX2 instructions.

https://www.intel.com/content/www/us/en/docs/cpp-compiler/developer-guide-reference/2021-8/intrinsics-for-avx-512-instructions.html