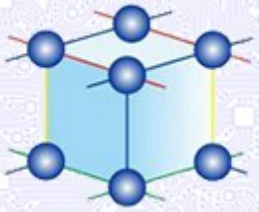


Сибирский
СуперКомпьютерный
Центр
ИВМиМГ СО РАН

ИВМиМГ

Оптимизация программы, реализующей метод частиц в ячейках для процессора Intel Xeon Phi 7290 KNL

А.В. Снытников

План доклада

- Решаемая задача: модель-алгоритм-программа
- Компиляция и запуск задачи на узлах с KNL
- Использование Intel Thread Checker (Intel Inspector) для поиска ошибок OpenMP-распараллеливания
- Использование Intel Advisor для поиска направлений оптимизации (векторизации) кода
- Использование Intel Vtune для обнаружения критичных по производительности участков кода

Решаемая задача: модель-алгоритм-программа

Основные уравнения

$$\frac{\partial f_{i,e}}{\partial t}+\vec{v}\frac{\partial f_{i,e}}{\partial \vec{x}}+\vec{F}\frac{\partial f_{i,e}}{\partial \vec{v}}=0$$

$$\nabla\times\vec{B}=4\pi\vec{j}+\frac{1}{c}\frac{\partial \vec{E}}{\partial t}$$

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

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

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

$$\vec{p}=\gamma\vec{v},\ \gamma^{-1}=\sqrt{1-v^2}$$

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

$$\vec{j}=\sum_{i,e}q_{i,e}\int f_{i,e}\vec{v}\,d\vec{v}$$

$$\rho=\sum_{i,e}q_{i,e}\int f_{i,e}\,d\vec{v}$$

- **Граничные условия:** периодические
- Требуется найти: зависимость f_e от времени

Начальные условия

→ρ_e = 1000, ρ_b = 1

$\rho = \rho_e+\rho_b$

→Импульсы электронов плазмы:
p_x,p_y,p_z — максвелловское распределение, $\sigma = T_e$ = 1.0

$$f=\exp\left(\frac{-p^2}{\sigma}\right)$$

→Импульсы ионов плазмы:0

→Импульс электронов пучка:
p_x = 50 p_y = p_z = 0

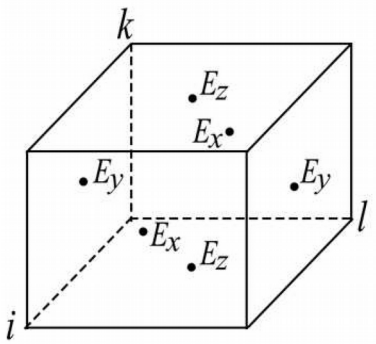
Аномальная электронная теплопроводность

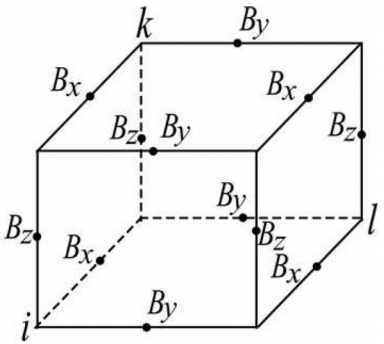
- В экспериментах на установке ГОЛ-3 (ИЯФ СО РАН) вследствие релаксации мощного электронного пучка наблюдается понижения электронной теплопроводности
- Коэффициент электронной теплопроводности уменьшается в 10^2-10^3 раз по сравнению с классическим значением для плазмы с такой плотностью и температурой
- Это позволяет лучше нагревать плазму и дольше удерживать ее в нагретом состоянии вследствие намного меньшего теплового потока на стенки установки

Эйлеров этап метода частиц в ячейках: **вычисление полей**

Схема Ленгдона-Лазинского

$$\frac{B^{m+1/2}-B^{m-1/2}}{\tau}=-\mathrm{rot}_h E^m$$
$$\frac{E^{m+1}-E^m}{\tau}=\mathrm{rot}_h B^{m+1/2}-j^{m+1/2}$$





$$\rho=\sum q_p \nu_p^{m+1/2}\overline{R}(x_p,x_i)$$

$$\frac{\rho^{m+1}-\rho^m}{\tau}+\mathrm{div}_h j^{m+1/2}=0$$

$$\mathrm{div}_h B=\frac{B_{x,i+1/2,k,l}-B_{x,i-1/2,k,l}}{h_x}+\frac{B_{y,i,k+1/2,l}-B_{y,i,k-1/2,l}}{h_y}+\frac{B_{z,i,k,l+1/2}-B_{z,i,k,l-1/2}}{h_z}$$

$$\mathrm{rot}_h B=\begin{pmatrix}\frac{B_{z,i,k,l-1/2}-B_{z,i,k-1,l-1/2}}{h_y}-\frac{B_{y,i,k-1/2,l}-B_{y,i,k-1,l-1}}{h_z}\\ \frac{B_{x,i-1/2,k,l}-B_{x,i-1/2,k,l-1}}{h_z}-\frac{B_{z,i,k,l-1/2}-B_{z,i-1,k,l-1/2}}{h_x}\\ \frac{B_{y,i,k-1/2,l}-B_{y,i-1/2,k,l}}{h_x}-\frac{B_{x,i-1/2,k,l}-B_{x,i-1/2,k-1,l}}{h_x}\end{pmatrix}$$

PIC-ядро

$$R(x)=\begin{cases}1-\frac{|x|}{h}, & |x|<h\\ 0, & |x|>h\end{cases}$$

Вшивков В.А. и др.,
Вычислительные технологии, Том 6, № 2, 2001.

Эйлеров этап метода частиц в ячейках: **движение частиц**

В методе частиц в ячейках среда разбивается на модельные частицы, траекториями движения которых являются характеристики кинетического уравнения Власова

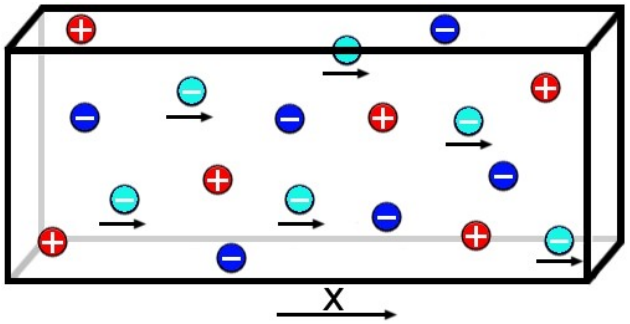
$$\frac{\partial p_{i,e}}{\partial t} = \kappa\left(E + [v, B]\right),$$

$$\frac{\partial r_{i,e}}{\partial t} = v_{i,e}.$$

$$p_{i,e} = \frac{v_{i,e}}{\sqrt{1 - v_{i,e}^2}}, \quad \kappa_e = -1, \quad \kappa_i = m_e / m_{i.}$$

Вшивков В.А. и др.,
Вычислительные технологии, Том 6, № 2, 2001.

Основные параметры



$$x \in [0, L], \quad y, z \in [0, n h_x]$$
$$k = 2\pi / L \longrightarrow W \sim e^{2\gamma t}, \gamma = \frac{1}{2}\frac{\partial \ln W}{\partial t}$$

$$f(v) = \frac{1}{\Delta v \sqrt{2\pi}} \exp - \frac{(v - v_0)^2}{2\Delta v^2}$$

n_b – плотность пучка

$2(\Delta v)^2$ – температура пучка

Гидродинамический режим $(k \Delta v \ll \gamma)$ $n_b = 2 \cdot 10^{-3}, \ \Delta v = 0.035$

Переходный режим $n_b = 10^{-3}, \ \Delta v = 0.14$

Кинетический режим $(k \Delta v \gg \gamma)$ $n_b = 2 \cdot 10^{-4}, \ \Delta v = 0.14$

Счетные параметры:

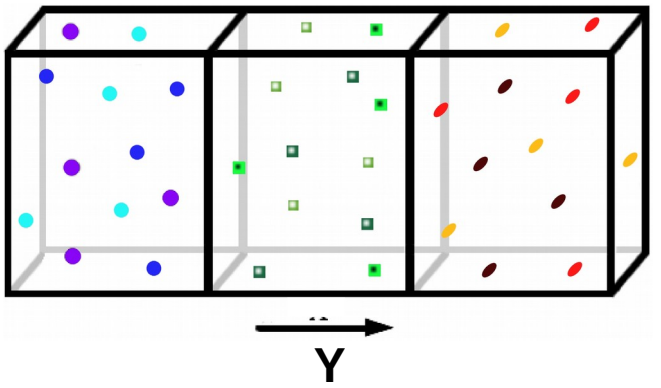
Длина области $L = 1.2566, \ 1.1424$

Сетка по пространству 100x4x4

Временной шаг $\tau = 0.001$

Число частиц в ячейке $lp = 50...20000$

Число процессоров $np = 16...256$



Заголовок цикла по частицам в программе UMKA, распараллеленного с помощью OpenMP

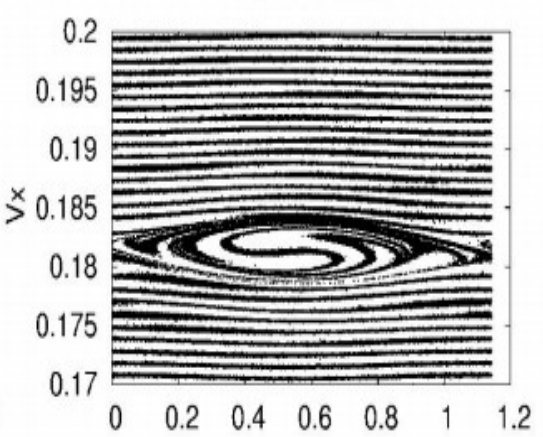
```
!$omp parallel private(x,y,z,x1,y1,z1,pu,pv,pw,i,l,k,s1,s2,s3,s4,s5,
!$omp+         s11,s21,s31,s41,s51,s61,i1,l1,k1,s6,x2,y2,z2,
!$omp+         s_i,s_l,s_k,s_l1,s_k1,s_i1,m,i2,l2,k2,
!$omp+         u,v,w,ps,sx,sy,sz,bx,by,bz,s,pu1,pv1,pw1,su,sv,sw)
!$omp+         firstprivate(h1,h2,h3,nt,ami,jm)
!$omp+         reduction(+:tpx,tpy,tpz)


      real_omp_threads = omp_get_num_threads()
      !iend1 px(1,3,1) ',meh,px(1,1,1),px(1,3,1)
      ! endif
      if(deb.ge.2) then
!$omp do
      do j=1,jm       ! do 100
      x=xi(j)
      y=yi(j)
      z=zi(j)
      pu=ui(j)
      pv=vi(j)
      pw=wi(j)
```

Верификация параллельной программы

- **Проблема:** сравнение параллельной программы с исходным последовательным вариантом по
  - средним величинам (импульса, энергии частиц, токов)
  - по ансамблю частиц (мат.ожидание, дисперсия скорости)
  - полной энергии поля
  - другим интегральным величинам
- **НЕ ГАРАНТИРУЕТ**, что программа
  - распараллелена правильно
  - в дальнейшем эта программа не даст какое-то совершенно другое (неправильное?!?) решение.
- Как минимум требуется проводить полное тестирование вычислительного алгоритма в параллельном варианте
- для каждой новой конфигурации процессоров и декомпозиции области

Верификация: результат

- Применение такой методики сравнения позволило,
- сопоставив параллельный и последовательный вариант
  - в течение 5 шагов (t = 0.005)
  - по 250 атрибутам
  - для каждой частицы
- в дальнейшем получить полное совпадение по всем частицам
  - до момента времени t = 120.0
  - при моделировании **двухпотоковой** неустойчи
    кинетическом режиме
- при использовании
  - **3D декомпозиции**
  - более **500 параллельных процессов**



Компиляция и запуск задачи на узлах с KNL

Загруженные модули

```
 module list
Currently Loaded Modulefiles:
  1) intel_license         3) profilers/advisor/2017.1.3.510716     5) profilers/tac/intel64/2017.3.030
  2) compilers/intel/2017.4.196 4) profilers/amplifier/2017.3.0.510739
  6) parallel/mpi.intel.knl/2017.4.196
```

Компиляция (Makefile)

```
LIB=-L/Compiler/11.1/038/lib/intel64/
FLAGS=-fopenmp -mcmodel large -shared-intel -g
#FLAGS= -g -fbounds-check -fopenmp -mcmodel=large
LDFLAGS= -fopenmp -mcmodel large -g
#LDFLAGS= -fopenmp -mcmodel=large -g
CC=gcc
MPICC=mpiifort
LD=mpiifort
```

```
mpiifort -fopenmp -mcmodel large -shared-intel -g   -c  para.f
mpiifort -fopenmp -mcmodel large -shared-intel -g   -c  out3d.f
…
```

Компиляция и **запуск** задачи на узлах с KNL

knl.sh:

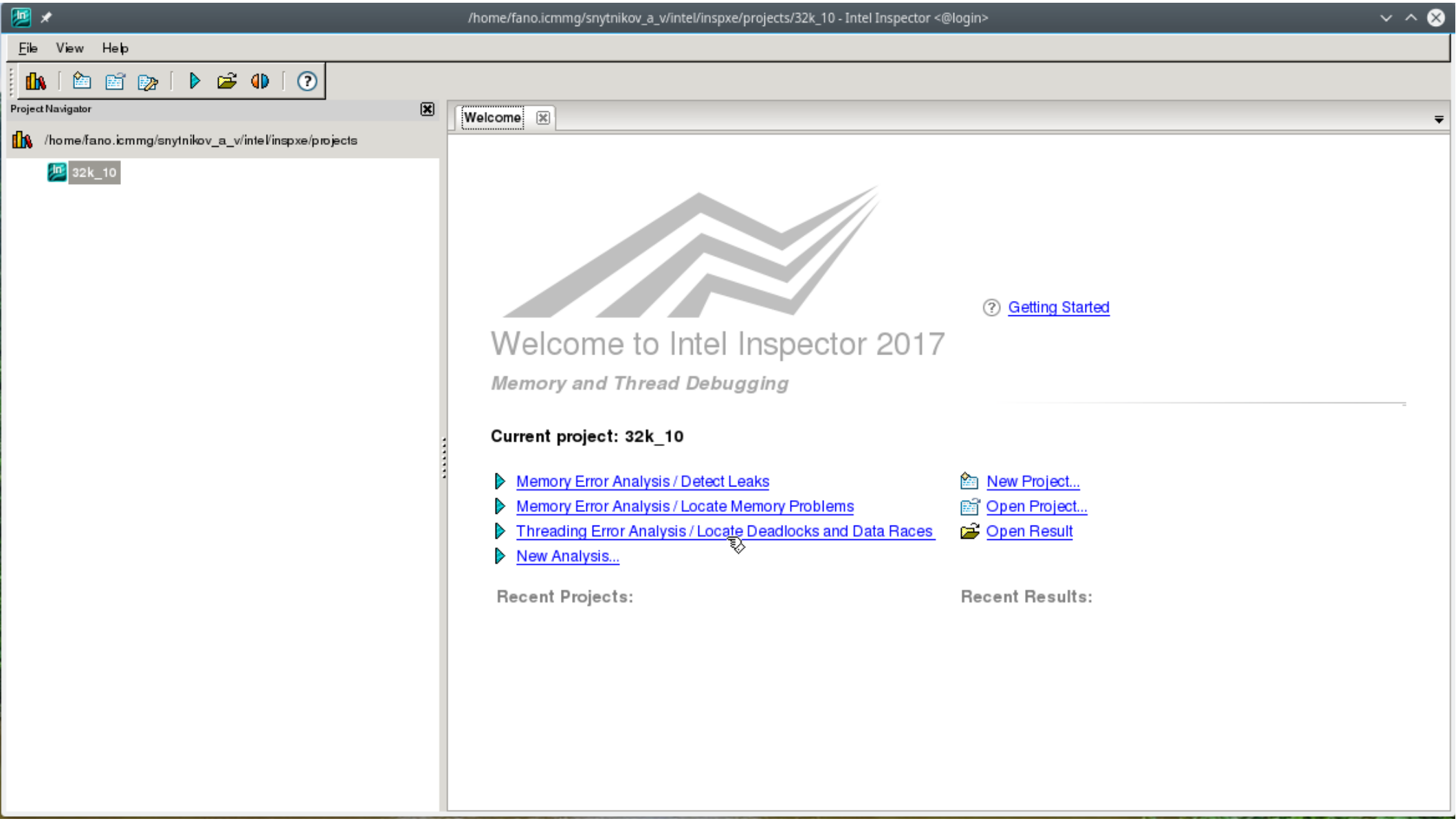
```
#!/bin/sh

# set the number of nodes
#SBATCH --nodes=1

# hyperthreading off
#SBATCH --threads-per-core=1



# set max wallclock time
#SBATCH --time=6-0

# set name of job
#SBATCH --job-name=KNL_ADV_SURVEY

# set queue name
#SBATCH -p knl
#SBATCH --ntasks-per-node=1



# run the application

mpirun -np 1 ./th
```

Постановка в очередь:

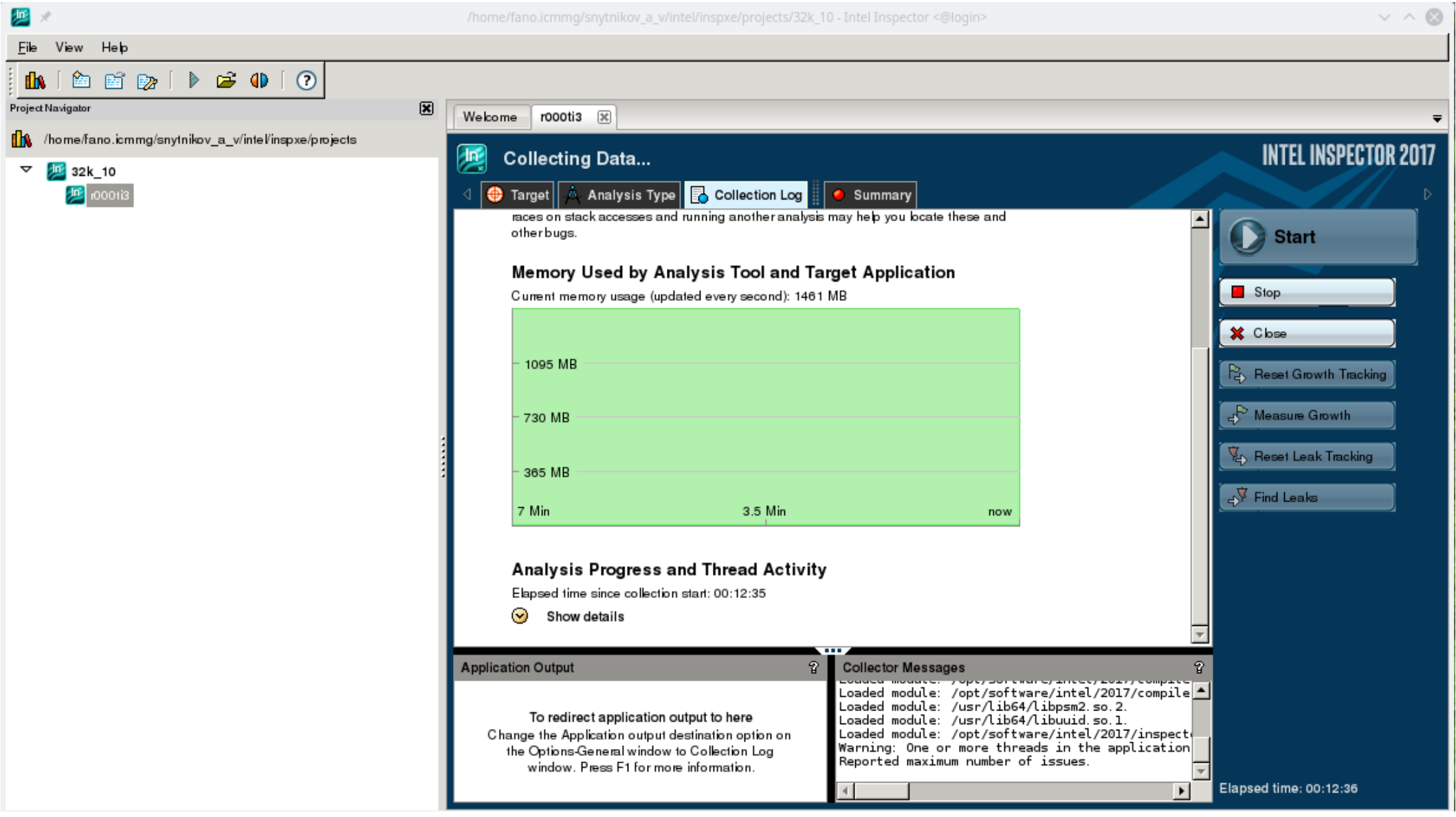
sbatch ./knl.sh

Использование Intel Thread Checker (Intel Inspector) для поиска ошибок OpenMP-распараллеливания

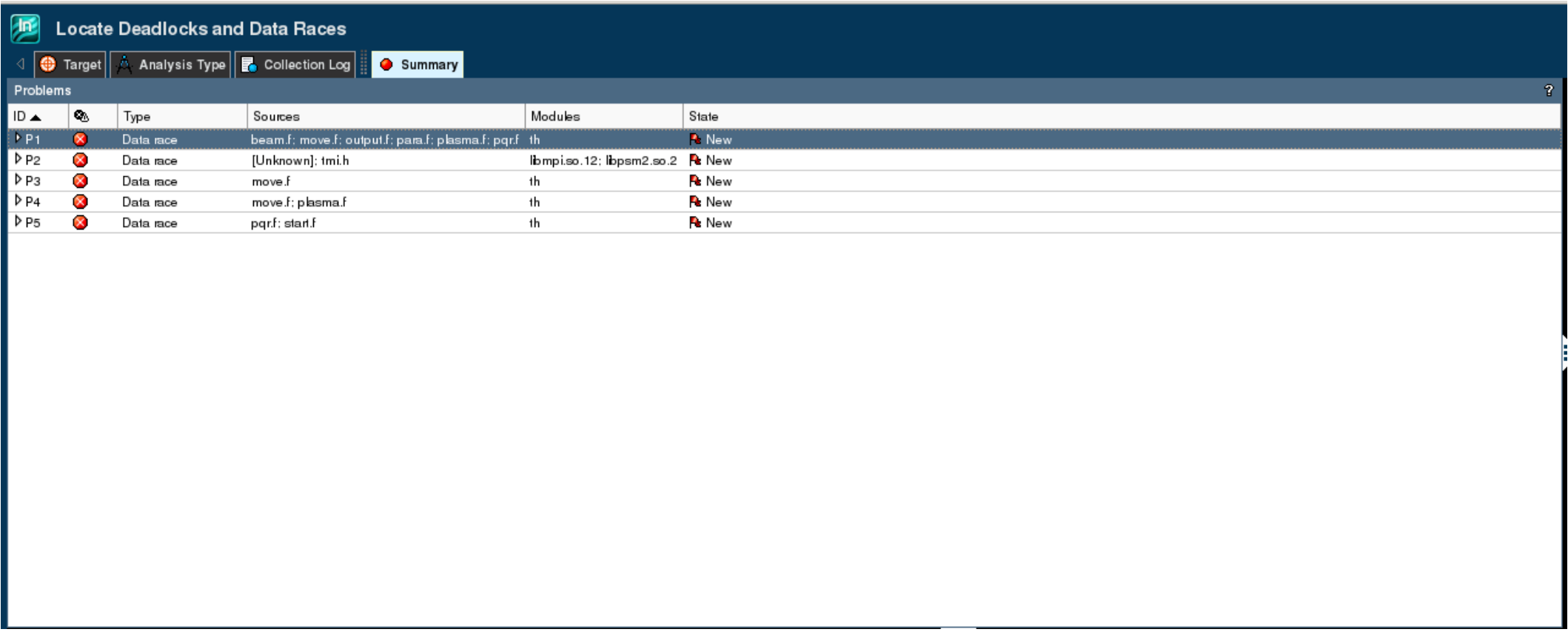
Стартовый экран



Процесс сбора данных



Результат: 5 ошибок типа data race



Ошибка 1: исходный текст
(доступен при компиляции с ключом -g)

Description	Source	Function	Module	Variable
Read	beam.f:46	beamboundcheck	th	block

```
44
45        if(baddst.eq.1) then
46           write(36,*) 'in beamboundcheck nt ',nt
47        endif
48
```

```
libifcoremt.so.5!for_write_seq_lis_xmit
th!beamboundcheck - beam.f:46
th!move_mod_mp_move3_$omp$parallel@175 - move.f:695
```

Write	beam.f:46	beamboundcheck	th	block

```
44
45        if(baddst.eq.1) then
46           write(36,*) 'in beamboundcheck nt ',nt
47        endif
48
```

```
libifcoremt.so.5!for_write_seq_lis_xmit
th!beamboundcheck - beam.f:46
th!move_mod_mp_move3_$omp$parallel@175 - move.f:695
```

Allocation site	para.f:33	parainit	th	block

```
31
32        if(baddst.eq.1) then
33          open(36,file=st1,form='formatted')
34        endif
35
```

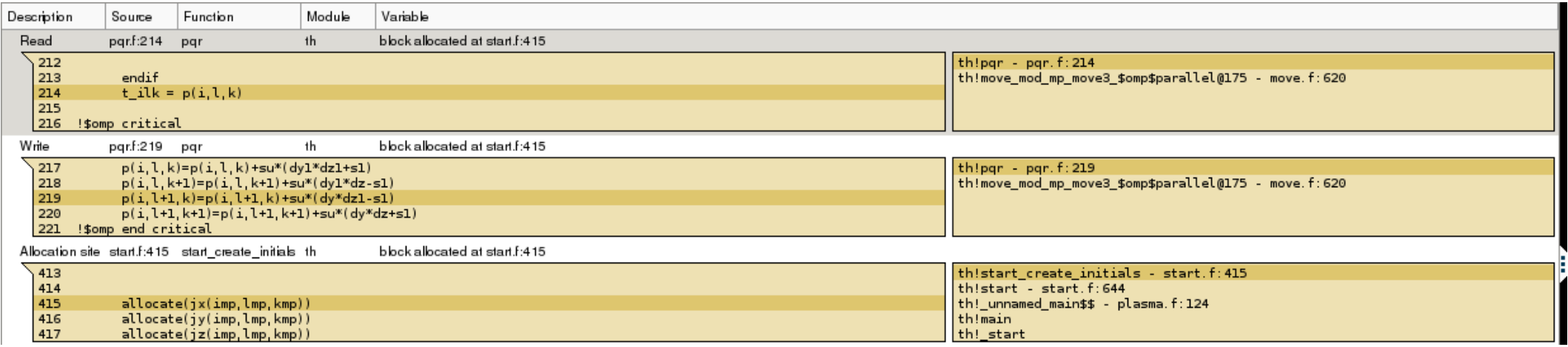
```
libifcoremt.so.5!for_open
th!parainit - para.f:33
th!_unnamed_main$$ - plasma.f:99
th!main
th!_start
```

HINT: Synchronization allocation site	pqr.f:217	pqr	th	block

```
215
216  !$omp critical
217        p(i,l,k)=p(i,l,k)+su*(dy1*dz1+s1)
218        p(i,l,k+1)=p(i,l,k+1)+su*(dy1*dz-s1)
219        p(i,l+1,k)=p(i,l+1,k)+su*(dy*dz1-s1)
```

```
th!pqr - pqr.f:217
th!move_mod_mp_move3_$omp$parallel@175 - move.f:620
```

Ошибка 2: исходный текст



Description	Source	Function	Module	Variable
Read	pqr.f:214	pqr	th	block allocated at start.f:415

```
212
213         endif
214         t_ilk = p(i,l,k)
215
216   !$omp critical
```

```
th!pqr - pqr.f:214
th!move_mod_mp_move3_$omp$parallel@175 - move.f:620
```

Write	pqr.f:219	pqr	th	block allocated at start.f:415

```
217         p(i,l,k)=p(i,l,k)+su*(dyl*dzl+sl)
218         p(i,l,k+1)=p(i,l,k+1)+su*(dyl*dz-sl)
219         p(i,l+1,k)=p(i,l+1,k)+su*(dy*dzl-sl)
220         p(i,l+1,k+1)=p(i,l+1,k+1)+su*(dy*dz+sl)
221   !$omp end critical
```

```
th!pqr - pqr.f:219
th!move_mod_mp_move3_$omp$parallel@175 - move.f:620
```

Allocation site	start.f:415	start_create_initials	th	block allocated at start.f:415

```
413
414
415         allocate(jx(imp,lmp,kmp))
416         allocate(jy(imp,lmp,kmp))
417         allocate(jz(imp,lmp,kmp))
```

```
th!start_create_initials - start.f:415
th!start - start.f:644
th!_unnamed_main$$ - plasma.f:124
th!main
th!_start
```

Запуск задачи с Intel Inspector на узлах с KNL

knl.sh:

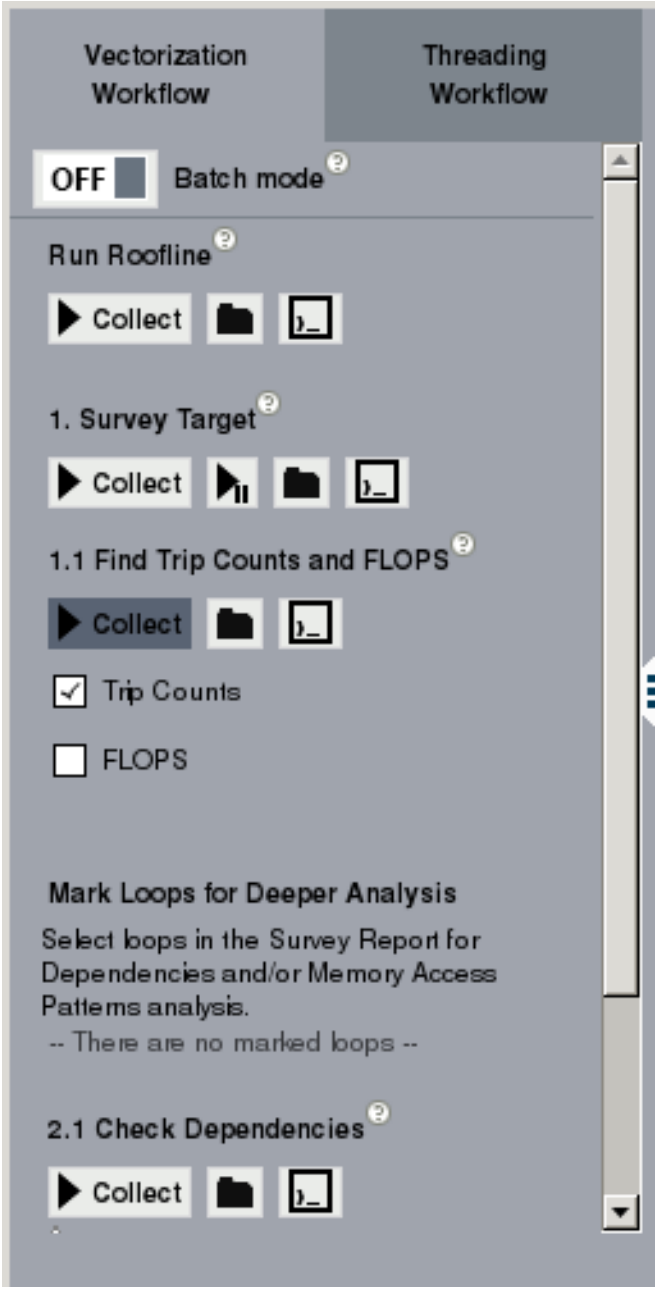
```
#!/bin/sh

# set the number of nodes
#SBATCH --nodes=1

# hyperthreading off
#SBATCH --threads-per-core=1



# set max wallclock time
#SBATCH --time=6-0

# set name of job
#SBATCH --job-name=KNL_ADV_SURVEY

# set queue name
#SBATCH -p knl
#SBATCH --ntasks-per-node=1



# run the application

mpirun -np 1 inspxe-cl -collect ti2 ./th
```

Постановка в очередь:

sbatch ./knl.sh

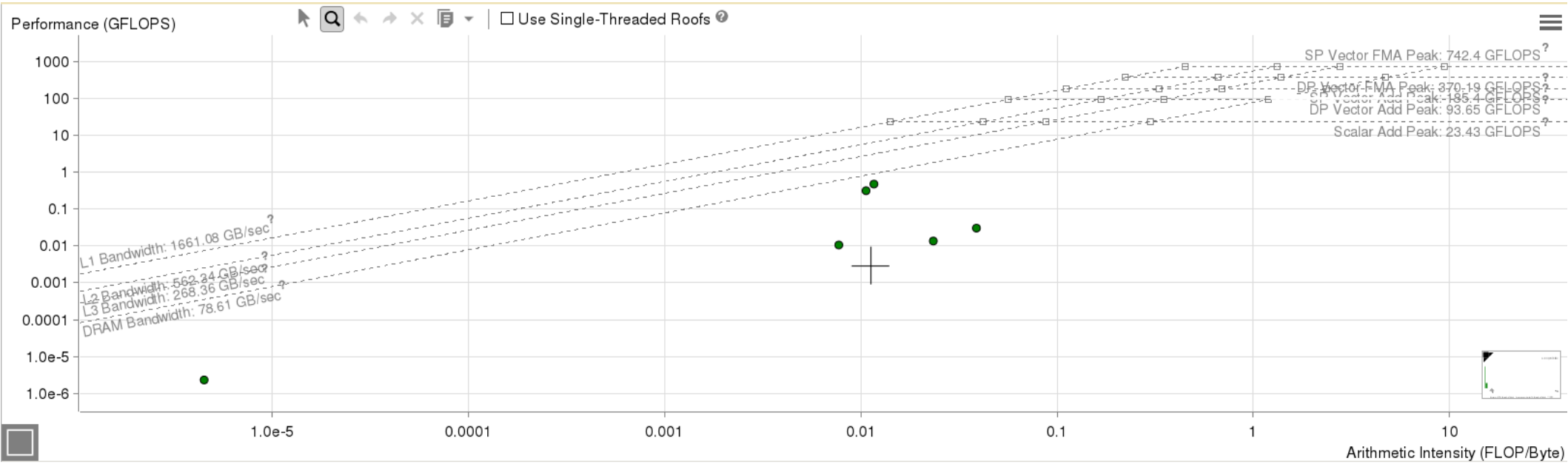
Использование Intel Advisor для поиска направлений оптимизации (векторизации) кода

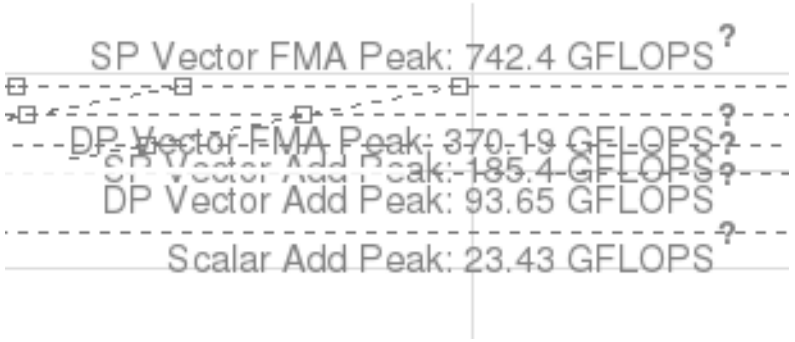
Выбор типа анализа в графическом интерфейсе



Анализ предельно достижимой производительности

roofline analysis





Рекомендации по доработке кода в целом

All known issues with all possible recommendations: C++ / Fortran

Issue: Potential underutilization of FMA instructions

Your current hardware supports the AVX2 instruction set architecture (ISA), which enables the use of fused multiply-add (FMA) instructions. Improve performance by utilizing FMA instructions.

Recommendation: Target the AVX2 ISA

Confidence: Low

Although static analysis presumes the loop may benefit from FMA instructions available with the AVX2 ISA, no AVX2-specific code executed for this loop. To fix: Use the `xCORE-AVX2` compiler option to generate AVX2-specific code, or the `axCORE-AVX2` compiler option to enable multiple, feature-specific, auto-dispatch code generation, including AVX2.

Windows* OS	Linux* OS
/QxCORE-AVX2 or /QaxCORE-AVX2	-xCORE-AVX2 or -axCORE-AVX2

Read More:

- ax, Qax; x, Qx
- Code Generation Options in the Intel® Fortran Compiler 16.0 User and Reference Guide
- Compiling for the Intel® Xeon Phi™ processor x200 and the Intel® AVX-512 ISA and Vectorization Resources for Intel® Advisor Users

Наиболее времяемкие циклы (Xeon 2630)

Program metrics

Elapsed Time	25.43s		
Vector Instruction Set	None	Number of CPU Threads	10
Total GFLOP Count	0.07	Total GFLOPS	0.00
Total Arithmetic Intensity	0.00		

Loop metrics

Metrics	Total	
Total CPU time	45.74s	100.0%
Time in scalar code	45.74s	100.0%

Vectorization Gain/Efficiency (Not available)

Top time-consuming loops

Loop	Self Time	Total Time	Trip Counts
[loop in sort at step.f:11]	2.500s	2.500s	7999
[loop in move_mod_mp_move3_ompparallel@175 at move.f:193]	0.570s	18.623s	1069
[loop in writefieldarrays at para.f:1960]	0.020s	0.330s	5
[loop in writeparticlelist at para.f:2010]	0.020s	1.700s	10669
[loop in move_mod_mp_move3_ompparallel@175 at move.f:719]	0.010s	0.010s	5

Collection details

Platform information

CPU Name	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
Frequency	2.55 GHz
Logical CPU Count	40
Operating System	Linux
Computer Name	login

Наиболее времяемкие циклы (Xeon Phi 7290)

Program metrics

Elapsed Time	53.39s		
Vector Instruction Set	None	Number of CPU Threads	11

Loop metrics

Metrics	Total	
Total CPU time	129.48s	100.0%
Time in scalar code	129.48s	100.0%

Vectorization Gain/Efficiency (Not available)

Top time-consuming loops

Loop	Self Time	Total Time
[loop in sort at step.f:11]	11.440s	11.440s
[loop in move_mod_mp_move3_ompparallel@175 at move.f:193]	3.760s	82.880s
[loop in writeparticlelist at para.f:2010]	0.100s	10.520s
[loop in parareducep at para.f:423]	0.020s	0.020s
[loop in emh1 at field.f:106]	0.020s	0.020s

Collection details

Platform information

CPU Name	Intel(R) Xeon Phi(TM) CPU 7290 @ 1.50GHz
Frequency	1.55 GHz
Logical CPU Count	288
Operating System	Linux
Computer Name	n01p006

Сравнение

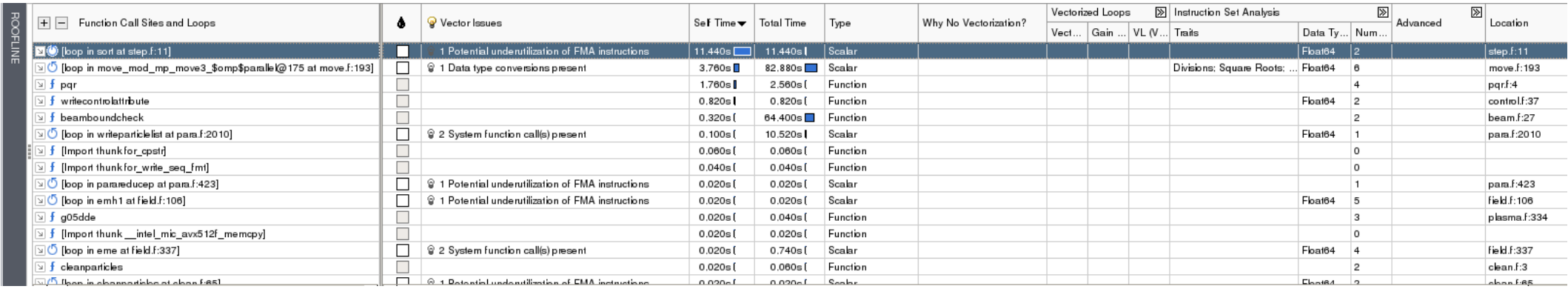
Top time-consuming loops

Loop	Self Time	Total Time	Trip Counts
[loop in sort at step.f:11]	2.500s	2.500s	7999
[loop in move_mod_mp_move3_ompparallel@175 at move.f:193]	0.570s	18.623s	1069
[loop in writefieldarrays at para.f:1960]	0.020s	0.330s	5
[loop in writeparticlelist at para.f:2010]	0.020s	1.700s	10669
[loop in move_mod_mp_move3_ompparallel@175 at move.f:719]	0.010s	0.010s	5

Phi

Loop	Self Time	Total Time
[loop in sort at step.f:11]	11.440s	11.440s
[loop in move_mod_mp_move3_ompparallel@175 at move.f:193]	3.760s	82.880s
[loop in writeparticlelist at para.f:2010]	0.100s	10.520s
[loop in parareducep at para.f:423]	0.020s	0.020s
[loop in emh1 at field.f:106]	0.020s	0.020s

Вопросы векторизации циклов на KNL



ROOFLINE

Function Call Sites and Loops		Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops			Instruction Set Analysis			Advanced	Location
							Vect...	Gain ...	VL (V...	Traits	Data Ty...	Num...		
[loop in sort at step.f:11]		1 Potential underutilization of FMA instructions	11.440s	11.440s	Scalar						Float64	2		step.f:11
[loop in move_mod_mp_move3_ompparallel@175 at move.f:193]		1 Data type conversions present	3.760s	82.880s	Scalar					Divisions; Square Roots; ...	Float64	6		move.f:193
pqr			1.760s	2.560s (	Function							4		pqr.f:4
writecontrolattribute			0.820s	0.820s (	Function						Float64	2		control.f:37
beamboundcheck			0.320s (	64.400s	Function							2		beam.f:27
[loop in writeparticlelist at para.f:2010]		2 System function call(s) present	0.100s (	10.520s	Scalar						Float64	1		para.f:2010
[Import thunk for_cpstr]			0.060s (	0.060s (	Function							0		
[Import thunk for_write_seq_fmt]			0.040s (	0.040s (	Function							0		
[loop in parareducep at para.f:423]		1 Potential underutilization of FMA instructions	0.020s (	0.020s (	Scalar							1		para.f:423
[loop in emh1 at field.f:106]		1 Potential underutilization of FMA instructions	0.020s (	0.020s (	Scalar						Float64	5		field.f:106
g05dde			0.020s (	0.040s (	Function							3		plasma.f:334
[Import thunk __intel_mic_avx512f_memcpy]			0.020s (	0.020s (	Function							0		
[loop in eme at field.f:337]		2 System function call(s) present	0.020s (	0.740s (	Scalar						Float64	4		field.f:337
cleanparticles			0.020s (	0.060s (	Function							2		clean.f:3

Запуск задачи с Intel Advisor на узлах с KNL

knl.sh:

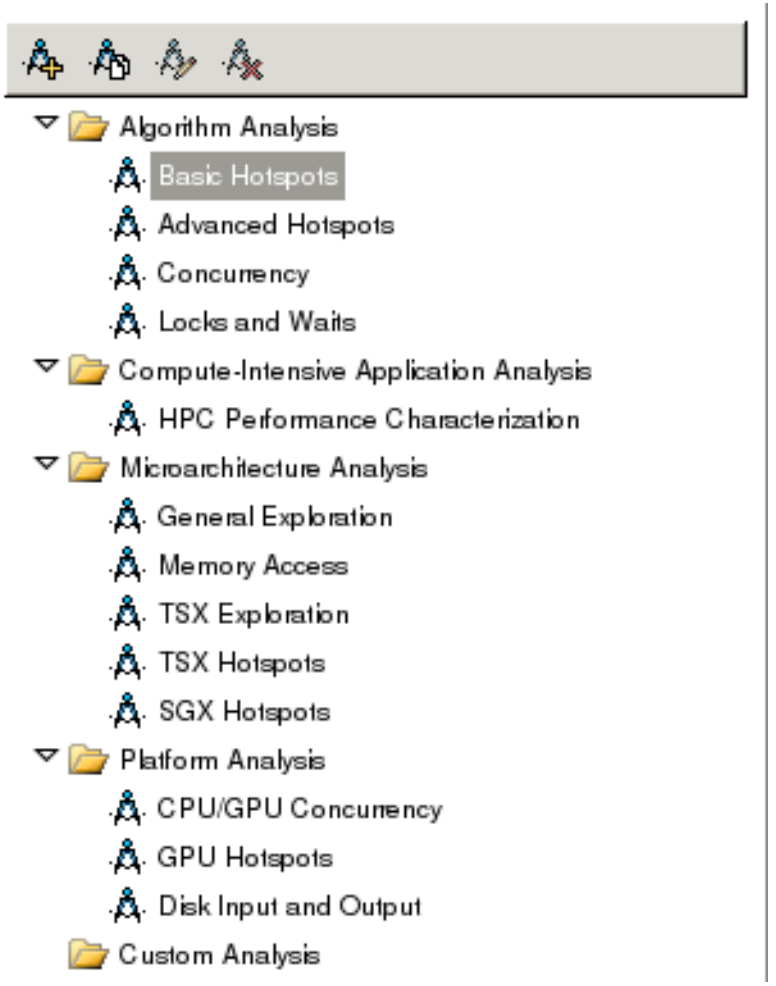
```
#!/bin/sh

# set the number of nodes
#SBATCH --nodes=1

# hyperthreading off
#SBATCH --threads-per-core=1



# set max wallclock time
#SBATCH --time=6-0

# set name of job
#SBATCH --job-name=KNL_ADV_SURVEY

# set queue name
#SBATCH -p knl
#SBATCH --ntasks-per-node=1



# run the application

mpirun -np 1 advixe-cl  -collect survey --project-dir ./survey ./th
```

Постановка в очередь:

sbatch ./knl.sh

Использование Intel Vtune для обнаружения критичных по производительности участков кода

Выбор типа анализа в графическом интерфейсе



Наиболее времяемкие части программы (Xeon 2630)

Elapsed Time: 24.887s

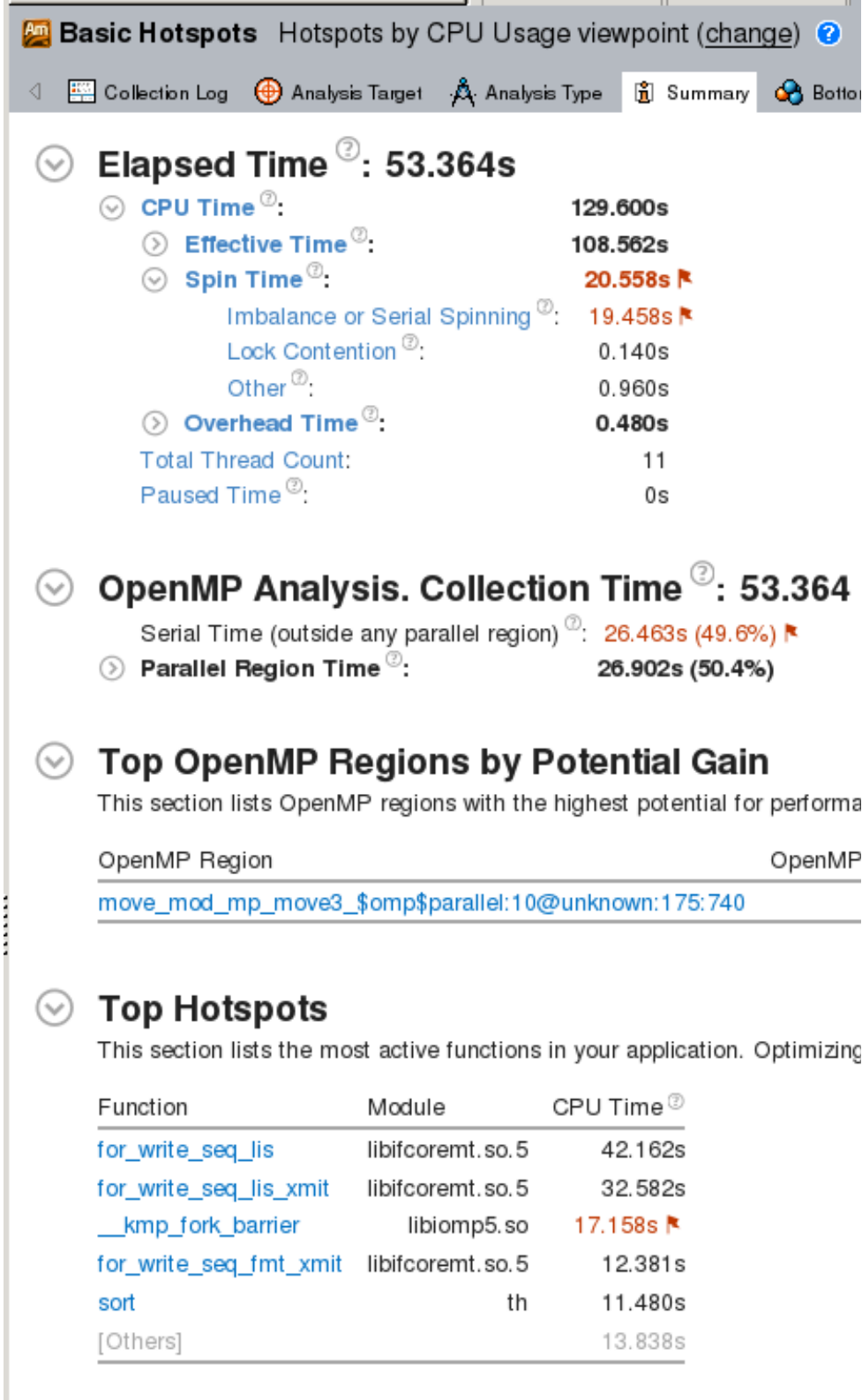
CPU Time:	44.880s
Effective Time:	23.470s
Spin Time:	21.350s
Imbalance or Serial Spinning:	20.760s
Lock Contention:	0.030s
Other:	0.560s
Overhead Time:	0.060s
Total Thread Count:	11
Paused Time:	0s

Top Hotspots

This section lists the most active functions in your application. Optimizi

Function	Module	CPU Time
__kmp_fork_barrier	libiomp5.so	15.120s
for_write_seq_lis	libifcoremt.so.5	10.570s
for_write_seq_lis_xmit	libifcoremt.so.5	7.480s
__kmpc_barrier	libiomp5.so	6.190s
for_write_seq_fmt_xmit	libifcoremt.so.5	2.120s
[Others]		3.400s

Наиболее времяемкие части программы (KNL)



Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Botto

Elapsed Time: 53.364s

CPU Time:	129.600s
Effective Time:	108.562s
Spin Time:	20.558s
Imbalance or Serial Spinning:	19.458s
Lock Contention:	0.140s
Other:	0.960s
Overhead Time:	0.480s
Total Thread Count:	11
Paused Time:	0s

OpenMP Analysis. Collection Time: 53.364

Serial Time (outside any parallel region):	26.463s (49.6%)
Parallel Region Time:	26.902s (50.4%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performa

OpenMP Region	OpenMP
move_mod_mp_move3_ompparallel:10@unknown:175:740	

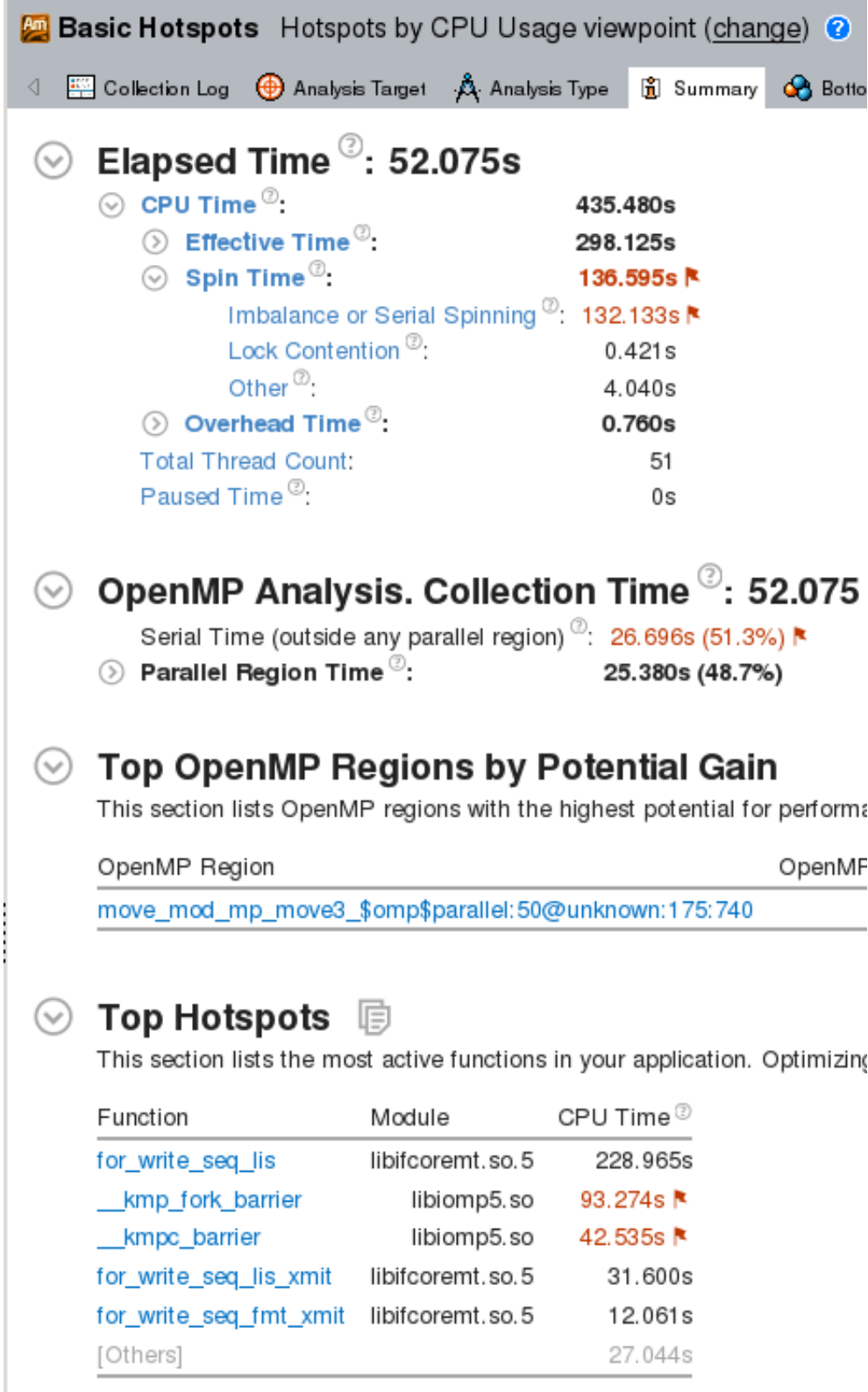
Top Hotspots

This section lists the most active functions in your application. Optimizing

Function	Module	CPU Time
for_write_seq_lis	libifcoremt.so.5	42.162s
for_write_seq_lis_xmit	libifcoremt.so.5	32.582s
__kmp_fork_barrier	libiomp5.so	17.158s
for_write_seq_fmt_xmit	libifcoremt.so.5	12.381s
sort	th	11.480s
[Others]		13.838s

Наиболее времяемкие части программы (KNL)

50 потоков OpenMP



Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Botto

Elapsed Time: 52.075s

CPU Time:	435.480s
Effective Time:	298.125s
Spin Time:	136.595s
Imbalance or Serial Spinning:	132.133s
Lock Contention:	0.421s
Other:	4.040s
Overhead Time:	0.760s
Total Thread Count:	51
Paused Time:	0s

OpenMP Analysis. Collection Time: 52.075

Serial Time (outside any parallel region):	26.696s (51.3%)
Parallel Region Time:	25.380s (48.7%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performa

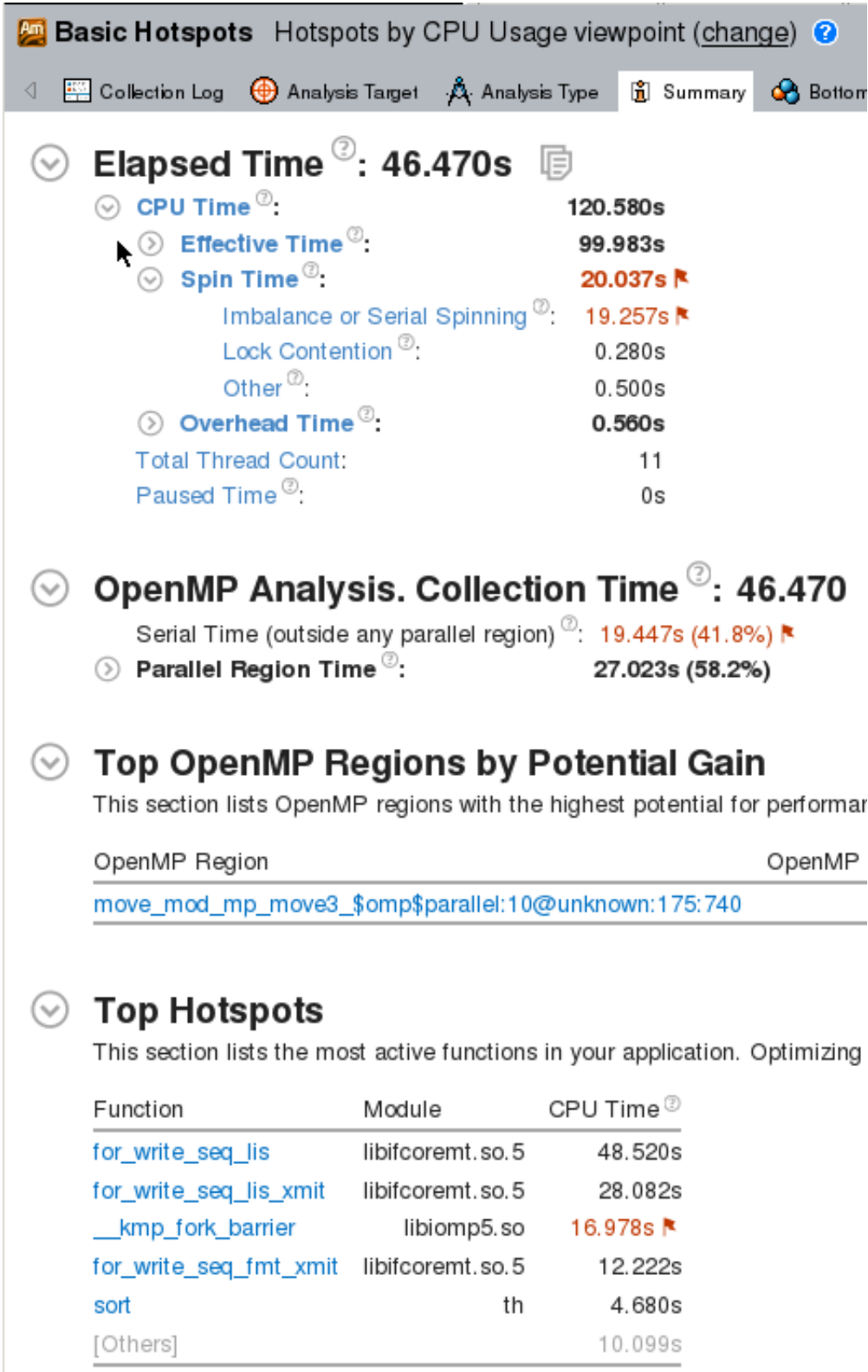
OpenMP Region	OpenMP
move_mod_mp_move3_ompparallel:50@unknown:175:740	

Top Hotspots

This section lists the most active functions in your application. Optimizing

Function	Module	CPU Time
for_write_seq_lis	libifcoremt.so.5	228.965s
__kmp_fork_barrier	libiomp5.so	93.274s
__kmpc_barrier	libiomp5.so	42.535s
for_write_seq_lis_xmit	libifcoremt.so.5	31.600s
for_write_seq_fmt_xmit	libifcoremt.so.5	12.061s
[Others]		27.044s

Наиболее времяемкие части программы (KNL) **оптимизация -xMIC-AVX512**



Анализ ограничений производительности

HPC Performance Characterization HPC Performance Characterization viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up

Elapsed Time: 485.042s

CPU Utilization: 0.2%

Average CPU Usage: 0.525 Out of 288 logical CPUs

Serial Time: 97.585s (20.1%)

Parallel Region Time: 387.457s (79.9%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
move_mod_mp_move3_ompparallel:10@unknown:175:740	14.376s	3.0%	387.457s

CPU Usage Histogram

Back-End Bound: 2.3%

L2 Hit Bound: 0.4% of Clockticks

L2 Miss Bound: 15.8% of Clockticks

SIMD Instructions per Cycle: 0.002

FP Instruction Mix:

% of Packed SIMD Instr.: 37.3%

% of Scalar SIMD Instr.: 62.7%

Наиболее затратные участки кода

Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time ▼								
	Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time			Overhead Time				
		Imbalance or Serial Spinning	Lock Contention	Other	Creation	Scheduling	Reduction	Atomics	Other
__kmp_fork_barrier	0s	14.750s	0s	0.370s	0s	0s	0s	0s	0s
for_write_seq_lis	10.570s	0s	0s	0s	0s	0s	0s	0s	0s
for_write_seq_lis_xmit	7.480s	0s	0s	0s	0s	0s	0s	0s	0s
__kmpc_barrier	0s	6.000s	0s	0.190s	0s	0s	0s	0s	0s
for_write_seq_fmt_xmit	2.120s	0s	0s	0s	0s	0s	0s	0s	0s
sort	1.950s	0s	0s	0s	0s	0s	0s	0s	0s
move_mod_mp_move3_ompparallel@175	0.560s	0s	0s	0s	0s	0s	0s	0s	0s
pqr	0.330s	0s	0s	0s	0s	0s	0s	0s	0s
for_write_seq_fmt	0.170s	0s	0s	0s	0s	0s	0s	0s	0s
writecontrolattribute	0.040s	0s	0s	0s	0s	0s	0s	0s	0s
__intel_avx_rep_memset	0.040s	0s	0s	0s	0s	0s	0s	0s	0s
beamboundcheck	0.040s	0s	0s	0s	0s	0s	0s	0s	0s
for_allocate	0s	0s	0s	0s	0s	0s	0s	0s	0.030s

Запуск задачи с Intel VTune на узлах с KNL

knl.sh:

```
#!/bin/sh

# set the number of nodes
#SBATCH --nodes=1

# hyperthreading off
#SBATCH --threads-per-core=1



# set max wallclock time
#SBATCH --time=6-0

# set name of job
#SBATCH --job-name=KNL_ADV_SURVEY

# set queue name
#SBATCH -p knl
#SBATCH --ntasks-per-node=1



# run the application

mpirun -np 1 amplxe-cl -collect hotspots -knob analyze-openmp=true ./th
```

Текущая производительность плазменного кода

- Наиболее времяемкий цикл программы содержит расчет движения 32 тыс. Модельных частиц, что означает 8 Mflop.
- Этот цикл выполняется за 0.112 сек., таким образом **80 Mflops**.
- Для Nvidia Kepler было получено 3.23 Gflops (А.В.Снытников)
- Есть результат в 1 Tflops на KNC (Nakashima, 2015)