ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И УПРАВЛЕНИЕ

Катедра "Компютърни системи"

СПЕЦИАЛНОСТ: "КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ" - СТЕПЕН БАКАЛАВЪР

Софтуерен пакет „GADGET " за космически N-body/SPH симулации в областта на астрофизиката на суперкомпютър BlueGene/P

Софтуерният пакет GADGET е свободен софтуерен код под GNU General Public License за извършване на космически N-body/SPH симулации на паралелни системи с разпределена памет. Комуникационният му модел е имплементиран със стандартния MPI комуникационен интерфейс. Кодът може да работи без проблем на всички съществуващи суперкомпютри, включително на клъстери от работни станции и на индивидуални персонални компютри. Софтуерът GADGET изчислява гравитационните сили чрез модел на йерархично дърво и представлява течности чрез средствата на гладката хидродинамика на частици (smoothed particle hydrodynamics – SPH). Кодът може да се използва за изучаване на изолирани системи или за симулации, включващи космологично разширение на пространството. При всички видове симулации GADGET следва еволюцията на самогравитираща сблъскваща се системи от N тела, като се допуска опционно включване на газова динамика.

Особености на GADGET:

- Йерархично мултиполюсно разширение за гравитационните сили, базирано на геометрично осмично дърво.
- Опционален TreePM метод, където дървото се използва за разпространяващи се на късо разстояние гравитационни сили само докато се изчисляват сили, разпространяващи се на дълго разстояние.
- Едноскоростна параметризация на изкуствената скорост, предложена от Монахан.
- Индивидуални времеви стъпки за всички частици.
- Балансирана декомпозиция на товара и динамично обновяване на дърветата.
- Поддръжка на паралелен вход/изход и няколко различни изходни формати, включително HDF5.

Софтуерът GADGET е написан на стандартен ANSI C и използва стандартизирания MPI 1.0 комуникационен интерфейс. За правилната му работа

са необходими и следните библиотеки: GNU Scientific Library GSL и FFTW библиотеката (Fastest Fourier Transform in the West). Ако се използва йерархичният файлов формат HDF5 е необходимо да се компилира и неговата библиотека.

Данните за симулациите се намират в т. нар. параметрови файлове. Това всъщност са текстови файлове, съдържащи двойки тагове и стойности. За всеки параметър се определя отделен ред. Първо се записва името на параметъра (таг) и след това присвоената му стойност, отделени с празно разстояние. Възможно е добавянето на допълнителен текст зад зададената стойност на параметъра. Последователността на параметрите е произволна, но всеки се записва само по веднъж.

Конфигуриране, настройка, верификация на суперкомпютър BlueGene/P на софтуерния пакет GADGET

Инсталиране на GADGET2 на суперкомпютър BlueGene/P

Необходими библиотеки:

1. MPI среда (например MPICH2).
2. Библиотеката GSL може да бъде изтеглена от
http://www.gnu.org/software/gsl/
3. Библиотеката FFTW 2.x.x може да бъде изтеглена от
http://www.fftw.org/download.html

Софтуерът GADGET2 изисква версия 2.x.

4. Библиотеката за поддръжка на HDF5 файлов формат HDF5 може да бъде изтеглена от
http://www.hdfgroup.org/HDF5/
5. файловете szip и zlib могат да се изтеглят от
http://www.hdfgroup.org/HDF5/release/obtain5.html
6. Софтуерният пакет GADGET2 може да се изтегли от
http://www.mpa-garching.mpg.de/gadget/

Установяване на работния каталог

За установяване на работната папка се изпълняват следните команди:

```
export GADGET_FOLDER="$HOME/Gadget2"
mkdir $GADGET_FOLDER
```

Чрез тези команди се настройва работната папка.

Променливата на средата $GADGET_FOLDER се използва по-късно при инсталацията.

В инструкциите за инсталиране се използват конкретни имена на библиотеки и техните версии, за по голяма яснота.

Копират се необходимите архиви tar в папката $GADGET_FOLDER (примерни имена):

```
gadget-2.0.7.tar.gz
fftw-2.1.5.tar.gz
```

```
gsl-1.9.tar.gz
hdf5-1.8.8.tar.gz
szip-2.1.tar.gz
zlib-1.2.5.tar.gz
```

Разархивира се всеки архив със следната команда:

```
tar -zxvf <file name>
```

Компилиране на софтуер на BlueGene/P

Компилирането на софтуер (приложения и библиотеки), който се изпълнява на изчислителните възли, става посредством cross-compiling, което означава, че се компилира на една машина, за друга различна машина: компилаторът работи на сървъра Front-End Node, но генерира код за изчислителните възли.

Системата разполага с два комплекта компилатори за `C`, `C++` и `FORTRAN`: `GNU Toolchain` и `IBM XL`.

За компилиране на софтуера `GADGET2` се използва компилаторът `IBM XL`. Също така той поддържа стандарта MPI.

Изходният код на приложение, което използва библиотеката MPI, може да се компилира на суперкомпютъра, като във файла `Makefile` или в скрипта `configure` се укаже компилатора, с който ще се използва (или по друг начин, в зависимост от самото приложение). Когато се изпълняват скриптове `./configure` е нужно да се укаже използването на компилатор `mpixlc`.

Въпреки, че има компилатор `gcc` на front-end възела (към който всеки потребител се свързва чрез протокола SSH), BlueGene/P има различна архитектура и софтуерът, който ще бъде изпълняван на него трябва да бъде компилиран с `mpixlc`.

Настройката за използването на компилатора `mpixlc` обикновено се извършва чрез настройване на променливата на средата `CC` с компилатор `mpixlc`. Например:

```
CC= mpixlc ./configure <parameters here>
```

Командата ще настрои променливата `CC` на компилатор `mpixlc` само за тази команда, след което ще я върне в начално състояние.

Още една особеност е, че компилаторът mpixlc по подразбиране компилира статични изпълними файлове.

При проблеми със свързването (linker), който се опитва да свърже динамична със статична библиотека, библиотеката се компилира като статична.

Инсталиране на библиотеката FFTW

Преминава се в каталога `fftw-2.1.5` и се изпълняват следните команди:

```
CC=mpicc ./configure --prefix=$GADGET_FOLDER/fftw-2.1.5-build --enable-
mpi --enable-shared=no --enable-type-prefix --enable-float
make
make install
```

При желание да се използва двойна точност на плаващата запетая, ключа `--enable-float`
се премахва от конфигурационния скрипт.

Единичната точност заема по-малко памет и изчисленията с нея стават по-бързо, но се ограничава точността.

Инсталиране на библиотеката HDF5

Инструкции за инсталацията може да бъдат намерени на следния линк:
`http://www.hdfgroup.org/ftp/HDF5/current/src/unpacked/release_docs/INSTALL`

Помощ за инсталация с паралелна поддръжка:
`http://www.hdfgroup.org/ftp/HDF5/current/src/unpacked/release_docs/INSTALL_parallel`

Помощ за съвместимост на API-то на библиотеката:
`http://www.hdfgroup.org/HDF5/doc/RM/APICompatMacros.html`

Библиотеката HDF5 използва zlib или szip библиотеки за компресия. Нужно е да се компилира поне една от тях (може и двете).

szip:

```
CC=mpicc ./configure --prefix=$GADGET_FOLDER/szip-2.1-build -enable-shared=no
make
make install
```

zlib:

```
CC=mpicc CFLAGS=-dynamic ./configure --prefix=$GADGET_FOLDER/zlib-1.2.5-build
make
make install
```

Забележка: `CFLAGS=-dynamic` е нужен, тъй като zlib може да бъде компилирана само като динамична библиотека.

HDF5

```
CC=mpicc ./configure --prefix=$GADGET_FOLDER/hdf5-1.8.8-build --enable-parallel --with-szlib=$GADGET_FOLDER/szip-2.1-build --with-zlib=$GADGET_FOLDER/zlib-1.2.5-build
make
make install
```

Премахва се една от 2-те библиотеки за компресия: `szip` или `zlib` от конфигурационните параметри.

Инсталиране на GSL

```
CC=mpicc ./configure --prefix=$GADGET_FOLDER/gsl-1.9-build --enable-shared=no
make
make install
```

Инсталиране на GADGET2

Преминава се в каталога `$GADGET_FOLDER/Gadget-2.0.7/Gadget2`.

В каталога се намират няколко `.h` и `.c` файлове, както и `Makefile`.

Отваря се файла `Makefile` за редактиране и се намира секцията с декларациите `"SYSTYPE"`.

Добавя се нова декларация:

```
SYSTYPE="BlueGeneBG"
```

Коментират се останалите декларации.

След тази секция следват дефинициите на тези декларации.

Добавя се нова дефиниция, като се коригират нужните пътища:

```
ifeq ($(SYSTYPE),"BlueGeneBG")
CC       =  mpicc
OPTIMIZE =  -O3 -Wall
GSL_INCL =  -I$(HOME)/Gadget2/gsl-1.9-build/include
GSL_LIBS =  -L$(HOME)/Gadget2/gsl-1.9-build/lib
FFTW_INCL=  -I$(HOME)/Gadget2/fftw-2.1.5-build/include
FFTW_LIBS=  -L$(HOME)/Gadget2/fftw-2.1.5-build/lib
MPICHLIB =
HDF5INCL =  -I$(HOME)/Gadget2/hdf5-1.8.8-build/include -DH5_USE_16_API
HDF5LIB  =  -L$(HOME)/Gadget2/hdf5-1.8.8-build/lib -lhdf5 -
L$(HOME)/Gadget2/zlib-1.2.5-build/lib -lz -L$(HOME)/Gadget2/szip-2.1-
build/lib/ -lsz
endif
```

Библиотеката HDF5 добавя параметъра `"-DH5_USE_16_API"`, който е необходим за съвместимост поради промени в API-то след версия 1.16.

В зависимост от това с каква библиотека за компресия е компилирана библиотеката HDF5, може да се премахне `szip` или `zlib`.

В началото на файла има различни настройки за компилиране, които добавят/премахват определени функции.

Намира се и се премахва коментара на следните 2 параметъра, ако е нужна двойна точност на плаващата запетая:

```
#------------------------------------ Single/Double Precision
OPT   +=  -DDOUBLEPRECISION
OPT   +=  -DDOUBLEPRECISION_FFTW
```

Това е нужно в случай че е компилиран ***FFTW*** с двойна точност (без флаг `--enable-float`).

За тестване на правилната конфигурация на библиотеката HDF5 се изтрива коментара на следния параметър:

```
#------------------------------------ Output
OPT   +=  -DHAVE_HDF5
```

Запазва се и се затваря файла. Изпълнява се следната команда:

```
make
```

Ако всичко е наред, в каталога се появява изпълним файл `Gadget2`.

Използване

Използването на GADGET2 може да се намери в:
`$GADGET_FOLDER/Gadget-2.0.7/Gadget2/users-guide.pdf`

Файлове с дефиниции на конфигурационни параметри за симулацията - parameterfile

Чрез parameterfile се контролират функциите на софтуерния код. Файлът трябва да се специфицира, когато се стартира кода. Всяка стойност на параметрите се дефинира чрез задаване на ключови думи, последвани от числова стойност или символен низ, разделени от празно пространство (интервали, табулации) от ключовата дума. Трябва да се използва отделен ред за всяка ключова дума, като стойността и ключовата дума се появяват на същия ред. Между ключовите думи може да се използва произволен брой празно пространство (включително празни редове).

Последователността на ключовите думи е случайна, но всяка дума трябва да се специфицира точно един път дори ако стойността не е от значение за симулацията (в противен случай се получава съобщение за грешка).

Редовете с водещ символ '%' се игнорират. Могат да бъдат добавени коментари в редовете с ключови думи, след уточняване на стойността за съответната ключова дума.

Ключови думи, маркирани с * могат да бъдат променяни по време на изпълнението на симулацията, т.е. промените на съответните стойности ще бъдат взети под внимание при стартиране на кода, като се използва флаг за рестартиране. Всички промени на други стойности на параметрите ще бъдат игнорирани.

Определяне на параметрите и съдържание на файла за симулация на сблъсък на две галактики:

```
%  Relevant files
InitCondFile         ICs/galaxy_bigendian.dat
OutputDir           galaxy/
EnergyFile          energy.txt
InfoFile            info.txt
TimingsFile         timings.txt
CpuFile             cpu.txt
RestartFile         restart
SnapshotFileBase    snapshot
OutputListFilename parameterfiles/output_list.txt

% CPU time -limit
TimeLimitCPU      36000  % = 10 hours
ResubmitOn        0
ResubmitCommand   my-scriptfile

% Code options
```

```
ICFormat                 1
SnapFormat               1
ComovingIntegrationOn    0
TypeOfTimestepCriterion  0
OutputListOn             0
PeriodicBoundariesOn     0

%  Caracteristics of run
TimeBegin           0.0        % Begin of the simulation
TimeMax                 3.0        % End of the simulation
Omega0                    0
OmegaLambda           0
OmegaBaryon           0
HubbleParam         1.0
BoxSize               0

% Output frequency
TimeBetSnapshot        0.5
TimeOfFirstSnapshot    0
CpuTimeBetRestartFile     36000.0    ; here in seconds
TimeBetStatistics         0.05
NumFilesPerSnapshot       1
NumFilesWrittenInParallel 1

% Accuracy of time integration
ErrTolIntAccuracy      0.025
CourantFac             0.15
MaxSizeTimestep       0.01
MinSizeTimestep       0.0

% Tree algorithm, force accuracy, domain update frequency
ErrTolTheta            0.5
TypeOfOpeningCriterion 1
ErrTolForceAcc         0.005
TreeDomainUpdateFrequency    0.1

%  Further parameters of SPH
DesNumNgb              50
MaxNumNgbDeviation     2
ArtBulkViscConst       0.8
InitGasTemp            0          % always ignored if set to 0
MinGasTemp             0

% Memory allocation
PartAllocFactor       1.5
TreeAllocFactor       0.8
BufferSize            25          % in MByte

% System of units
UnitLength_in_cm         3.085678e21        ;  1.0 kpc
UnitMass_in_g            1.989e43           ;  1.0e10 solar masses
UnitVelocity_in_cm_per_s 1e5                ;  1 km/sec
GravityConstantInternal  0
```

```
% Softening lengths
MinGasHsmlFractional 0.25
SofteningGas       0
SofteningHalo      1.0
SofteningDisk      0.4
SofteningBulge     0
SofteningStars     0
SofteningBndry     0
SofteningGasMaxPhys       0
SofteningHaloMaxPhys      1.0
SofteningDiskMaxPhys      0.4
SofteningBulgeMaxPhys     0
SofteningStarsMaxPhys     0
SofteningBndryMaxPhys     0
MaxRMSDisplacementFac 0.2
```

Подбор на тестови данни за симулация със софтуерния пакет GADGET

Сблъсък на галактики

Симулацията за сблъсък на две галактики осъществява движение на две дискови галактики една към друга, което води до сливането им. Всяка галактика се състои от звезден диск и масивна и разширена тъмна материя. Галактиките се състоят съответно от 20000 и 40000 частици. Данните се съхраняват в двоичен формат с разширение .dat.

Адиабатен колапс на газова сфера

Тази симулация разглежда гравитационен колапс на само-гравитираща сферата от газ при ниска температура, която първоначално има профил на плътност 1 / R. Газовата сфера пада към центъра под собственото си тегло, където отскача обратно със силна ударна вълна. Симулацията използва Нютоновата физика в естествена система от единици (G = 1). Това е малка симулация на 1472 частици.

Образуване на клъстер от галактики в космоса

Този проблем използва динамиката на сблъскването в разширяваща се вселена. Това е малка клъстерна симулация, която е създадена с генератор Zic с първоначални условия Bepi Tormen с използване на вакуумни граници. В симулацията участват общо 276 498 частици. В централната зона с висока резолюция има 140 005 частици, заобиколени от граничен регион на две нива – вътрешен, съдържащ 39 616 частици, и външен с 96,877 частици.

Мащабно формиране на структура, включваща газ

Симулацията се състои от 32000 частици тъмна материя и 32000 частици газ и проследява формирането на структура във вселената. Включена е адиабатна

физика на газа и минималната температура на газа е определена на 1000 К. Този пример използва мрежа от първоначални условия, където газовите частици са поставени в центровете на мрежата, очертана от частиците на тъмната материя. Симулацията започва със z = 10 и в резултат на изпълнението се получават файлове с моментни изображения – отместване 5, 3, 2, 1 и 0. Може да се прочете файл с моментна снимка на тази симулация, да се извличат газовите частици от него и да се начертае тяхното пространствено разпределение.

Експериментални резултати от симулации на софтуерния пакет GADGET на суперкомпютър BlueGene/P

За опитна постановка са използвани примерните симулации, придружаващи библиотеката Gadget2. Проведени са четири вида експериментални симулации със софтуерен пакет GADGET на суперкомпютър BlueGene/P:

- Мащабно формиране на структура в космоса от 32000 частици тъмна материя и 32000 частици газ.
- Сблъсък на две галактики в космоса, състоящи се от 40000 и 20000 частици.
- Образуване на клъстер от галактики в космоса, състояща се от 276498 частици.
- Адиабатен колапс на газова сфера, състоящи се от 1472 частици.

За входни данни са използвани дадените примерни такива: galaxy_bigendian.dat, gassphere.param, lcdm_gas.param, galaxy.param, cluster.param. Форматът е big endian, тъй като това е формата на подредба на байтовете, използван в архитектурата на BlueGene/P.

За параметри на симулацията са използвани дадените във файловете с разширение .param, като са променени стойностите на следните параметри:

- TimeMax - 2.0
- PartAllocFactor - 2.0
- TreeAllocFactor - 245.8

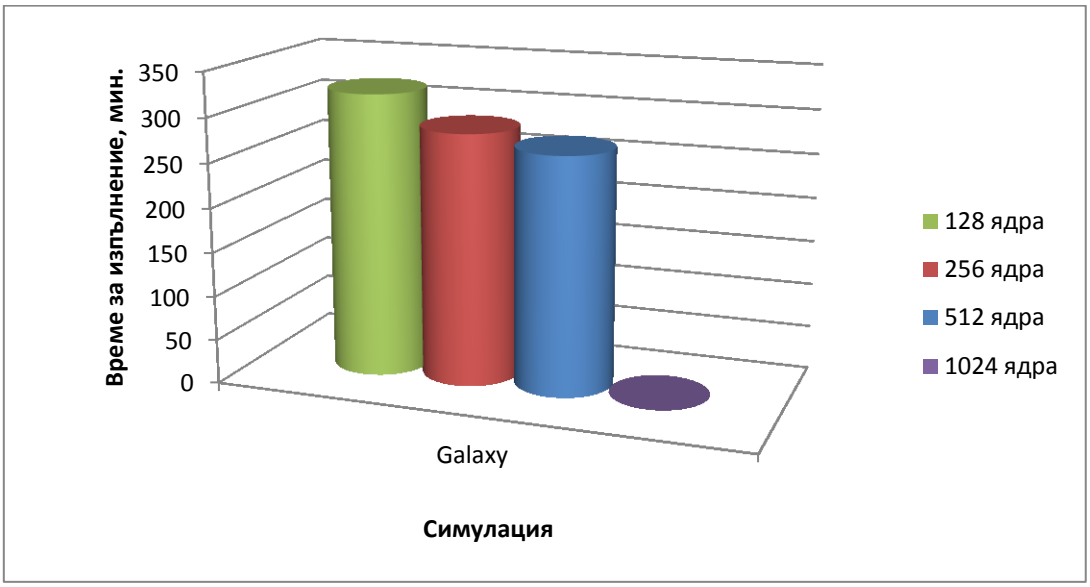
Коефициентите на PartAllocFactor и TreeAllocFactor е необходимо да бъдат променени, тъй като в противен случай при по-голям брой процеси работата не успява да бъде разпределена правилно между процесите и следва неуспех при изпълнение на задачата.

Софтуерът GADGET е изпълнен на 128, 256, 512 и 1024 ядра в режим VN на суперкомпютъра BlueGene/P. Направени за различни видове космически симулации с използване на описаните данни.

Проведени са симулации за сблъсък на две галактики в космоса, състоящи се от 40000 и 20000 частици, без включен инструмент за профилиране `Scalasca` при използване на следния брой ядра: 128, 256, 512, 1024. При последния вариант с 1024 ядра, симулацията не успя да разпредели правилно работата си и бе прекратена. Резултатите са показани в таблица 1 и на фиг. 1.

Таблица 1. Експериментални резултати за симулация на сблъсък на две галактики в космоса на суперкомпютър BlueGene/P

Симулация / Процеси	Време за изпълнение, минути			
	128 ядра	256 ядра	512 ядра	1024 ядра
Galaxy	320.9	284.43	268.52	-

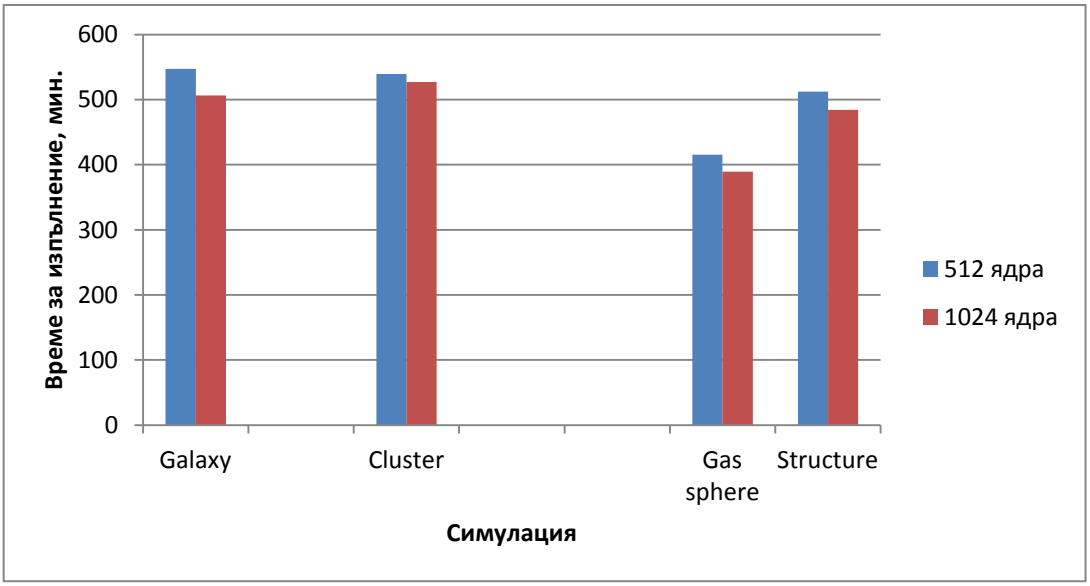


Фиг. 1. Времена на изпълнение за симулация на сблъсък на две галактики със софтуер Gadget на различен брой ядра на суперкомпютър BlueGene/P

Проведени са четири вида експериментални симулации с включен профилиращ инструмент Scalasca. Симулациите, наборите от данни и времената за изпълнение са дадени в Таблица 2 и на фиг. 2.

Таблица 2: Симулации, експериментални данни и времена за изпълнение на софтуер GADGET на 512 и 1024 ядра на суперкомпютър BlueGene/P

Симулация		Време за изпълнение, минути	
Dataset	Particles	512 ядра	1024 ядра
Galaxy	galaxy 1: 40000 galaxy 2: 20000	547.20	506.34
Cluster	(halo): 140005 (disk): 39616 (bulge): 96877	539.27	527.19
Gas sphere	gas: 1472	415.56	389.40
Structure	dark matter: 32000 gas: 32000	512.10	484.34



Фиг. 2. Сравнение на времената изпълнение на различните симулации при използване на различен брой ядра на суперкомпютър BlueGene/P

Профилиране на изпълнението на софтуерния пакет GADGET-2 с профайлер SCALASCA

Инструментът SCALASCA поддържа измерване и анализ на MPI програмни конструкции, най-широко използвани във високо мащабируемите приложения, създадени за високопроизводителни компютърни системи. Те са написани на програмните езици C, C++ и Fortran и са предназначени за широк кръг от високопроизводителни платформи. Инструментът SCALASCA се използва чрез командата *scalasca* с подходящите атрибути след нея.

По време на изпълнението на инструментирания код на паралелна компютърна система потребителят може да създаде обобщен отчет (профил), съдържащ общите показатели на производителността на индивидуалните пътеки за извикване на функции. Освен това, могат да се създадат проследявания на събития чрез записването на индивидуални събития, настъпили по време на изпълнението, от които по-късно може да се създаде профил или визуализация на времевата линия. Възможността за създаване на обобщен отчет от изпълнението на кода на програмата е полезна, тъй като може да се получи представа за получената производителност.

Когато е включено проследяването, всеки процес създава файл за проследяване, който съдържа записи на всички локални за него събития. След приключване на приложението, инструментът SCALASCA зарежда всички файлове за проследяване в основната памет и ги анализира паралелно, като използва толкова процесори, колкото е използвало и самото приложение. По време на анализа се търсят характерни модели, показващи чакащи състояния и свързани с производителността характеристики, след което откритите случаи се

класифицират по категория и се оценява тяхното значение. Резултатът от тези действия е създаването на отчет, съдържащ анализ на моделите, със структура, подобна на обобщения отчет, но включващ и показатели, свързани пряко с неефективната комуникация и синхронизация между по-високите нива. И двата отчета съдържат показатели на производителността за всяка пътека за извикване на функция и системен ресурс, които могат да се изследват интерактивно чрез графично приложение.

Команди на софтуерния инструмент SCALASCA

Използването на инструмента SCALASCA включва три етапа – програмна инструментация, измерване и анализ на изпълнението и изследване на отчета за извършения анализ. Командата *scalasca* обезпечава три действия, които извикват съответните команди *skin, scan* и *square*. Тези действия са:

scalasca –instrument

Използва се вмъкване в кода на приложението на повиквания към измервателната система на инструмента SCALASCA. Това може да стане автоматично, полуавтоматично или чрез връзка с библиотеки, които са предварително инструментирани.

scalasca –analyze

Използва се за контролиране на измервателната среда по време на изпълнение на приложението, както и за автоматичен анализ на проследяването (ако е включено) след прекратяване на приложението.

scalasca –examine

Използва се за последваща обработка на отчета за извършения анализ, създаван от обобщеното измерване по време за изпълнението на приложението, и/или анализ на проследяванията след приключване изпълнението на програмата, както и за стартиране на графичното приложение CUBE.

Компилиране и свързване

Командите за компилиране и свързване при създаване на приложение, което трябва да се анализира, трябва да съдържат пред себе си

scalasca –instrument

Колекция от измервания по време на изпълнението

Колекцията от измервания на инструмента SCALASCA по време на изпълнението и техният анализ са достъпни чрез използване на командата

scalasca –analyze,

която включва следните стъпки:

- настройки на измерването;
- изпълнение на приложението;

- колекция от измерените данни;
- автоматичен анализ на проследяването след прекратяване на приложението (ако е зададено).

Използва се уникален каталог за всеки измервателен експеримент, който не трябва да съществува при неговото стартиране. Измерването се прекратява, ако съществува определеният каталог. Създава се име по подразбиране за всеки каталог, което се състои от името на изпълняваното приложение, настройките на изпълняващата го компютърна система (брой на процесите, нишките и т.н.) и настройките на измерването. Името на каталога винаги започва с „*epik_*”, а неговото местоположение може точно да се определи в инструмента SCALASCA чрез опцията *„-e <path>”* или да се промени чрез настройките на променливите.

След завършване на измерването, каталогът *epik* съдържа различни файлови дневници, както и един или повече отчети за извършени анализи. По подразбиране обобщението от изпълнението на приложението се използва за създаването на обобщен отчет за броя на посещенията и изразходваното време при всяко извикване на път от всеки процес. При измерванията на функциите на MPI се включват времето и съобщенията, както и файловата входно-изходна статистика. В обобщения отчет също се включват хардуерни броячни показатели, ако са изискани.

Данните за проследяване на събитията също могат да се събират като част от измерването, като се създава файл за проследяване тип *epilog* за всеки процес. За събирането на тези данни като част от измерването трябва да се използва командата *scalasca –analyze –t*. В този случай експерименталният анализ на проследяването започва автоматично след завършване на измерването.

Командата за преглед на резултатите от измерванията е

scalasca –analyze –n

и може да се използва за визуализация (но не и действително изпълнение) на измерването и командите за стартиране на анализа. Възможно е и да се покаже допълнителен коментар на информацията (чрез *–v*), особено ако измерването или анализа са неуспешни.

Изследване на отчета за извършения анализ

Резултатите от автоматичния анализ се съхраняват в един или повече отчети в каталога *epik*. Тези отчети могат да се изследват чрез командата:

scalasca -examine epik_<title>

Последващата обработка се извършва при първото изследване на директорията, преди стартиране на графичното приложение CUBE3. Ако командата *scalasca –examine* се изпълни върху вече обработена директория или върху файл тип CUBE, зададен като аргумент, графичното приложение се стартира автоматично.

Кратък текстов отчет, без стартиране на графичното приложение може да се получи чрез командата:

scalasca -examine -s epik_<title>

Графичното приложение CUBE3 може да се стартира и чрез командите:

cube3 epik_<title>

cube3 <file>.cube

Графично приложение CUBE3

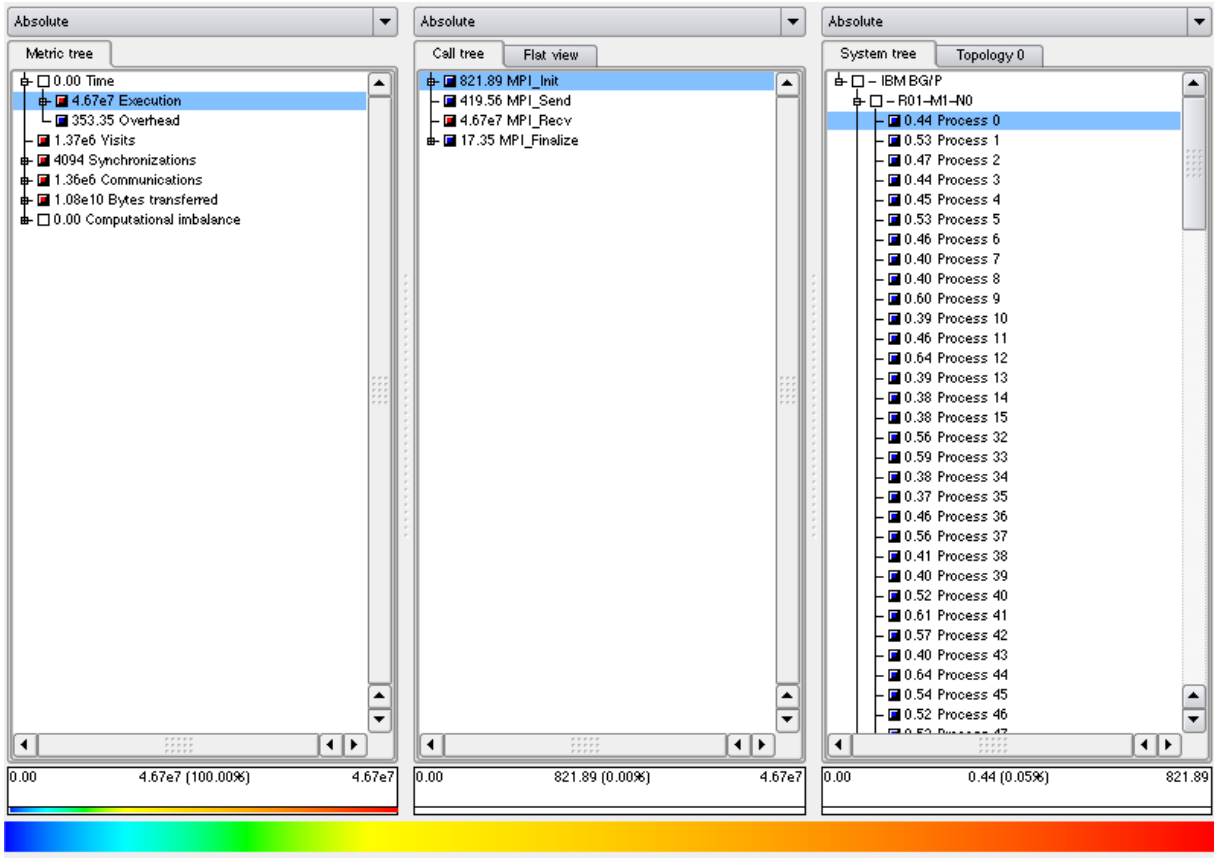
Графичното приложение CUBE (Cube Uniform Behavioral Encoding) е подходящо за изобразяване на широк кръг от данни за производителността на паралелни програми, използващи библиотеката MPI. Приложението позволява интерактивно изследване на данните за производителността по скалируем начин. Скалирането се постига по два начина – йерархично разделяне на различните измерения и тяхното обобщение. Всички показатели са равномерно разпределени на един и същи екран, като по този начин се осигурява възможност за лесно сравнение на ефектите от различните начини на изпълнение на дадено приложение.

Приложението CUBE е проектирано около съществуващ модел за данни от високо ниво, свързан с поведението на приложенията, наречен *кубично пространство на производителността (cube performance space)*. Това пространство се състои от три измерения – ***метрично, програмно и системно***. ***Метричното*** измерение съдържа множество от показатели, като време за комуникация и др. ***Програмното*** измерение съдържа дървото на повикванията за конкретното приложение, което обхваща всички извиквания на пътеки, на които може да се нанесат метричните стойности. ***Системното*** измерение съдържа всички елементи, които се изпълняват паралелно, без значение дали са процеси или нишки, в зависимост от паралелния програмен модел.

Всяко измерение от пространството на производителността е организирано йерархично. Първото измерение, а именно метричното, е организирано с включваща йерархия, като даден показател от по-ниско ниво е подмножество на своите родители. Например времето за комуникация е подмножество на общото време за изпълнение. Второто измерение, а именно програмното, има йерархична организация на дървото на повикванията. Въпреки това понякога може да бъде полезно абстрахирането от йерархията на дървото на повикванията, като например, ако някой се интересува от тежестта на даден метод, независимо от това откъде се извиква той. Третото и последно измерение, а именно системното, е организирано с няколко нива на йерархията – компютърна система, възел, процес и нишка.

Графичната компонента на CUBE има възможност да изобрази различните измерения на пространството на производителността, използвайки три

взаимосвързани дървовидни браузъри (фиг. 3). Те са свързани по такъв начин, че едното измерение може да се разгледа по отношение на другото. Връзката се базира на избор – във всяко дърво може да се избере един или повече възли.



Фиг. 3. Изобразяване на различните измерения на паралелната производителност чрез графичното приложение CUBE

CUBE 3.1 QT

Софтуерът за визуализация, който е инсталиран на суперкомпютъра IBM BlueGene/P, е CUBE 3.1 QT (фиг. 3). Той има три дървовидни браузъри, всеки от които представлява едно от измеренията на пространството на производителността. По подразбиране лявото дърво изобразява метричното измерение, средното дърво изобразява програмното измерение, а дясното дърво – системното измерение. Възлите в метричното дърво представляват показатели. Възлите в програмното измерение могат да имат различни значения, в зависимост от конкретно избрания изглед. Възлите в системното измерение представляват компютърни системи, възли, процеси или нишки отгоре надолу.

Всеки възел има стойност, която се нарича тежест и се изобразява едновременно с цифрова стойност и под формата на оцветен квадрат. Цветовете позволяват лесното разпознаване на интересни възли дори в големите дървета, докато цифровите стойности позволяват прецизното им сравнение. Стойността на даден възел може да се определи визуално чрез неговото оцветяване. Колкото

по-наляво в изобразената цветова скала се намира възелът, толкова неговата стойност е по-малка. И обратното, колкото по-надясно е в изобразената цветова скала, толкова и стойността му е по-голяма. По подразбиране цветовете в цветовата скала са заимствани от спектъра на светлината – започва се от синьо, след което се преминава към циан, зелено и жълто, докато се стигне до червено, като по този начин се обхващат всички възможни стойности на възлите. Стойностите, които са точно нула, се изобразяват с бял цвят или с минималния цвят, в зависимост от желанието на потребителя. Освен това, цветовите настройки може да се променят. В допълнение към това, потребителят може също да зададе и собствени стойности, които да отговарят на минималния и максималния цвят в цветовата скала. Тогава всички стойности извън този интервал ще се изобразяват със сив цвят.

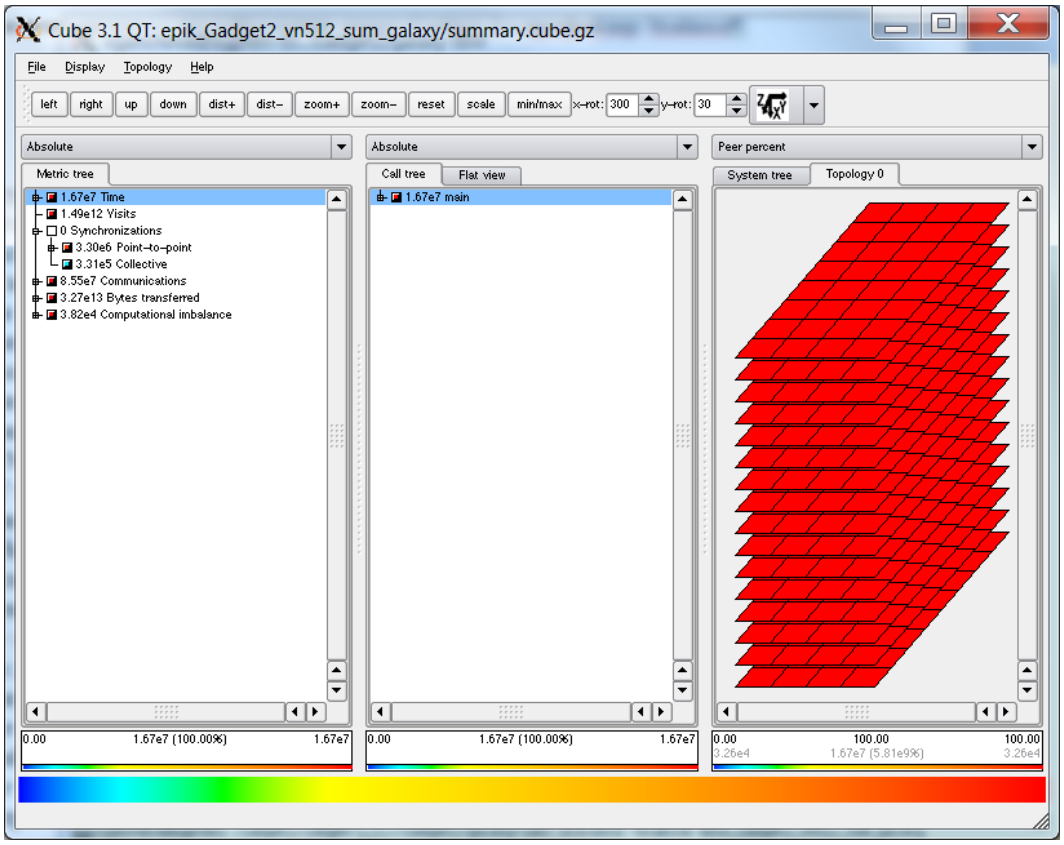
Под всеки от трите дървовидни браузъри има информационно поле. Ако няма заредени данни или не са избрани такива, полето остава празно. В противен случай полето показва по-детайлна информация за избраната стойност в дървото. Информационното поле и дървовидния браузър могат да имат различна точност на представяните данни, т.е. полето може да изобразява избраните стойности с повече знаци след десетичната запетая, отколкото самия дървовиден браузър.

Най-лявото число в информационното поле показва най-малката стойност в дървото, а най-дясното число – най-голямата стойност. Между тези две числа се намира текущата стойност на избрания възел, следвана от нейния процент в скалата между най-ниската и най-високата стойност, изобразени в двата края на полето.

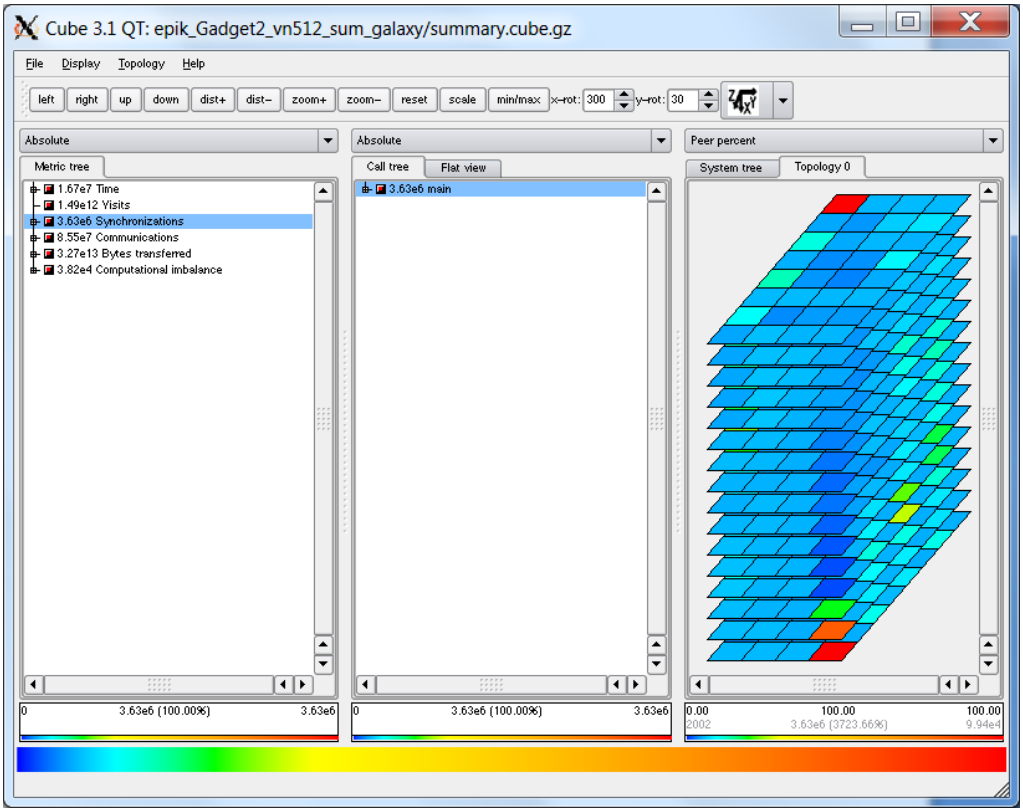
Малка цветна лента под числата в информационното поле показва позицията на цвета на избрания възел в общата скала на цветовете. В случай на неопределени стойности, лентата се запълва със сива мрежа.

Анализ на паралелната производителност на софтуер GADGET

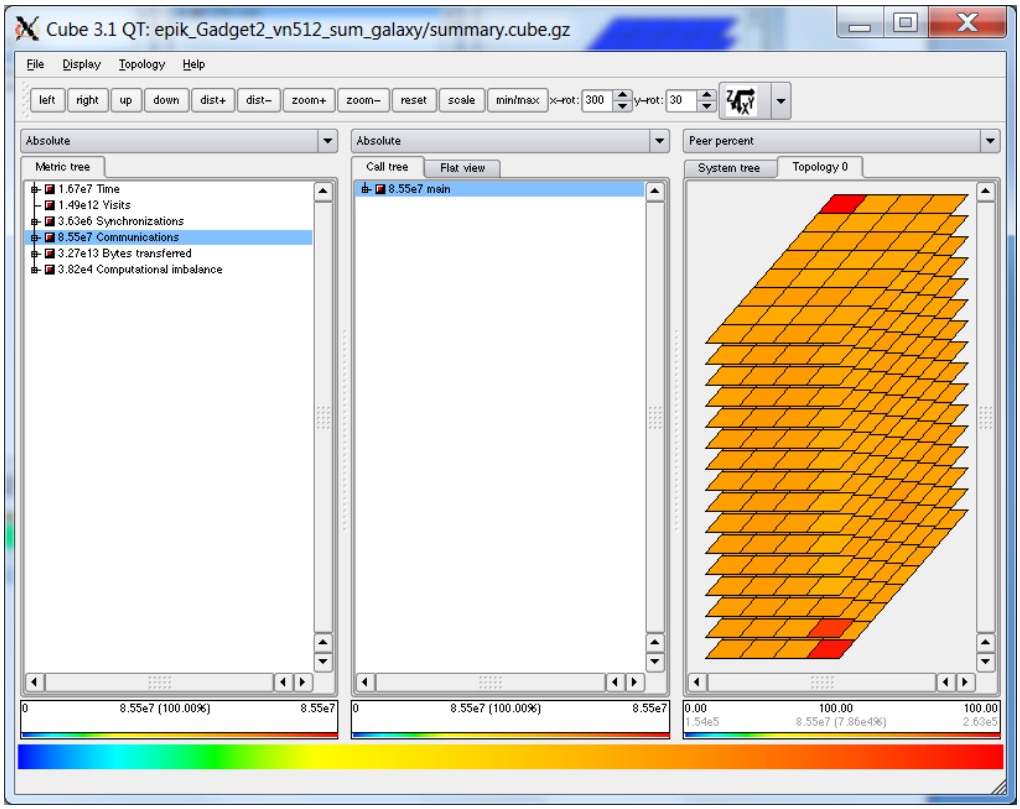
Направен е анализ на производителността на софтуер GADGET при изпълнението на симулацията за сблъсък на две галактики с помощта на графичното приложение CUBE 3.1 QT. Квадратите в прозореца „Topology" изобразяват всички ядра, които са заети с изпълнението на програмата (в случая 512). Оцветяването в еднакъв цвят на процесите (в случая в червено) показва, че всички са натоварени равномерно (фиг. 4). На фиг. 5 са дадени профилите при синхронизация, на фиг. 6 – при синхронизация, на фиг. 7 – при трансфер на данни. Профилите на фиг. 8 показват разпределението на изчислителния товар.



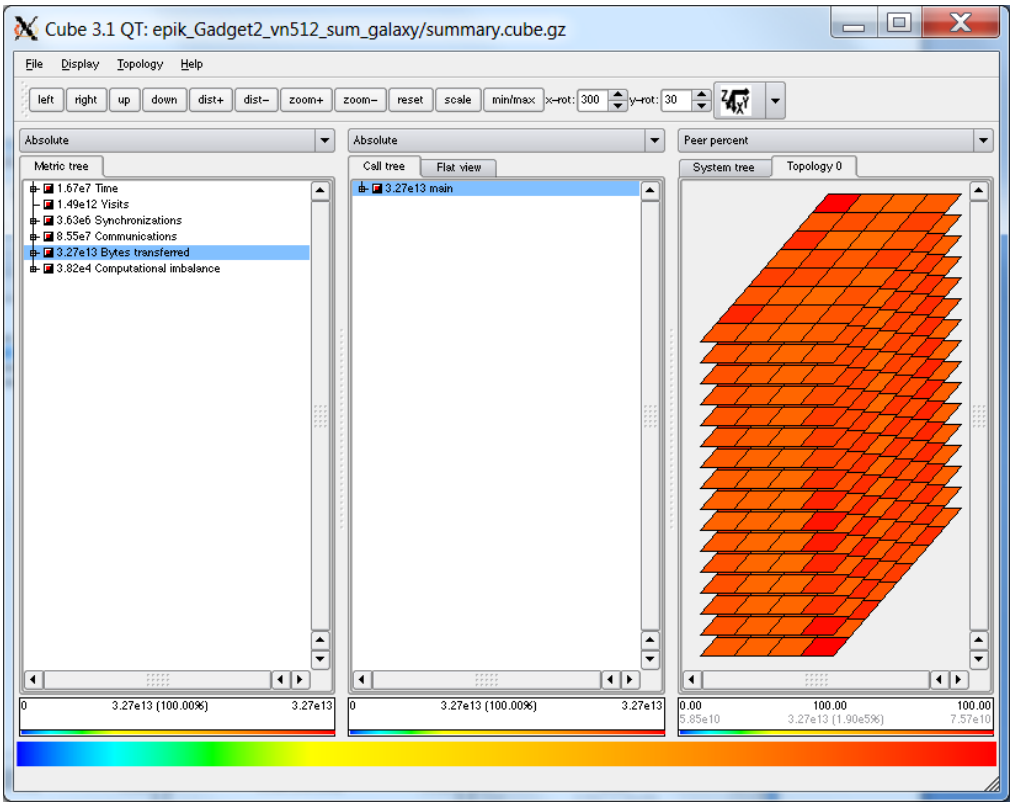
Фиг. 4. Изобразяване на профилите на натоварването на процесите по време на изпълнение на софтуера GADGET за симулация на сблъсък на две галактики на 512 ядра



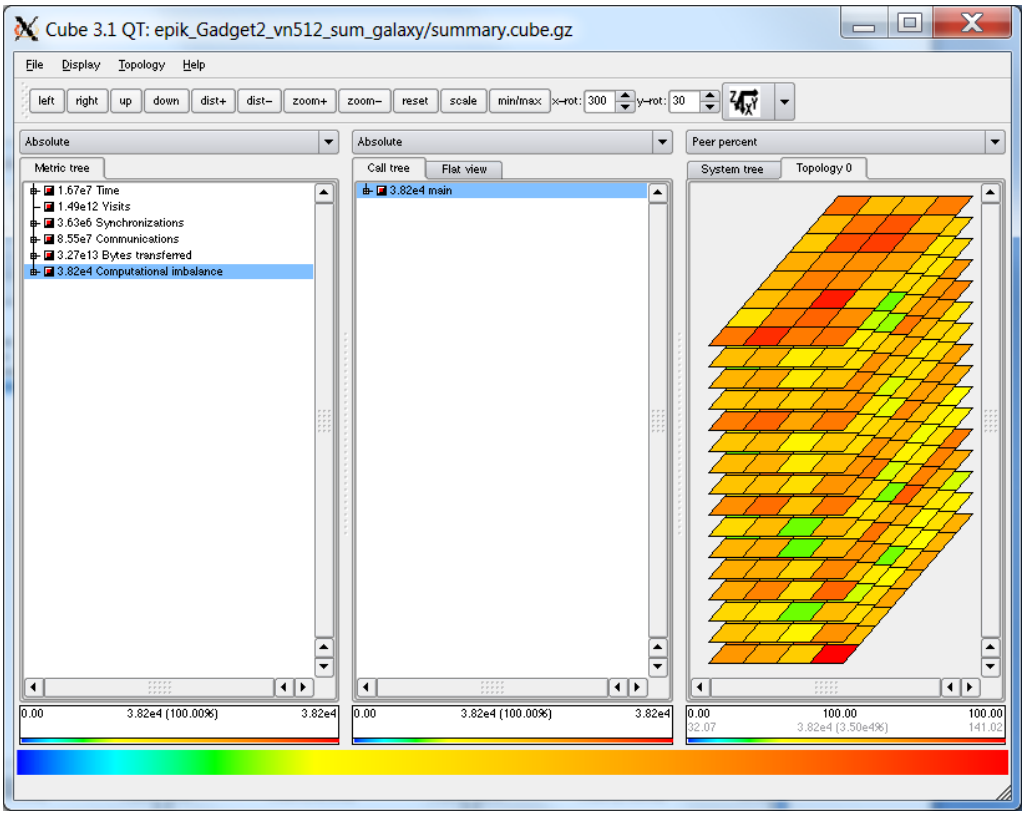
Фиг. 5. Изобразяване на профилите при синхронизация по време на изпълнение на софтуера GADGET за симулация на сблъсък на две галактики на 512 ядра



Фиг. 6. Изобразяване на профилите при комуникация по време на изпълнение на софтуера GADGET за симулация на сблъсък на две галактики на 512 ядра



Фиг. 7. Изобразяване на профилите при трансфер на данни по време на изпълнение на софтуера GADGET за симулация на сблъсък на две галактики на 512 ядра



Фиг. 8. Изобразяване на профилите за разпределение на изчислителния товар при изпълнение на GADGET за симулация на сблъсък на две галактики на 512 ядра

Визуализация на изходните данни от симулациите

Основният резултат от симулацията със софтуера GADGET-2 са моментни изображения (snapshot), които представляват състоянието на системата в определено време. GADGET-2 поддържа паралелен изход от разпределени изображения в няколко файла, като всеки от файловете се записва от група от процесори. Тази процедура позволява по-лесно обработване на много големи симулации, вместо да се обработва един файл с голям размер, например много по-лесно е да се обработват няколко файла с по-малки размери - от по няколкостотин MB, вместо един файл от няколко GB. Също така, времето за входно-изходните операции намалява, като се записват паралелно няколко файла.

Всяко моментно изображение се състои от k на брой файлове, където k се задава чрез променливата NumFilesPerSnapshot. За $k > 1$, имената на файловете се означават във формат XXX.Y за моментните изображения, където XXX е за броя на изображенията, Y е броя на файловете за изображение. Броят на моментните изображения се задава чрез променливата SnapshotFileBase.

Всеки от отделните файлове на даден набор от файлове за моментно изображение съдържа променлив брой на частиците. Въпреки това, всички файлове имат един и същи основен формат, както и всички файлове са в двоичен формат. Двоичното представяне на данните за частиците е за предпочитане, тъй като позволява много по-бързи входно-изходни операции, отколкото при

файловете във формат ASCII. Също така, файловете са много по-малки за съхранение на данните и загубите са по-малко.

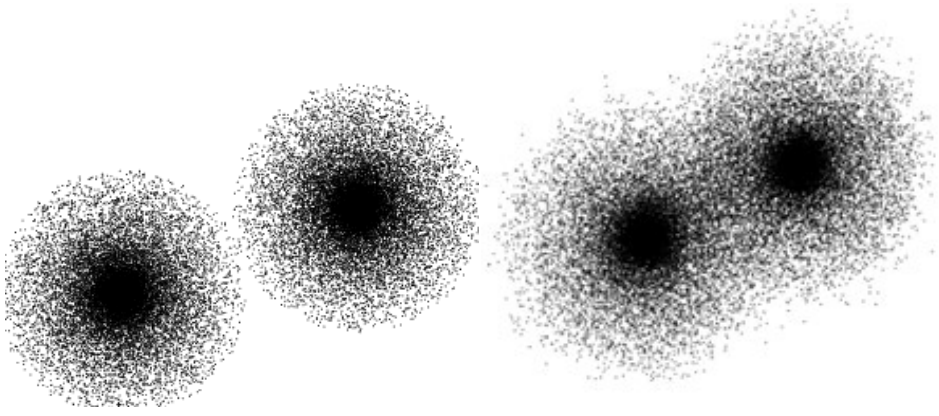
Данните във файловия формат по подразбиране на GADGET-2 (избран като fileformat 1 за SnapFormat) се организират в блокове, всеки съдържащ определена информация за частиците. Например, има блок за координатите, блок за температура и т.н. Първият блок има специална роля - съдържа глобална информация за набора на частиците, например броят на частици от всеки тип, броят на файловете, използвани за дадената снимка и др.

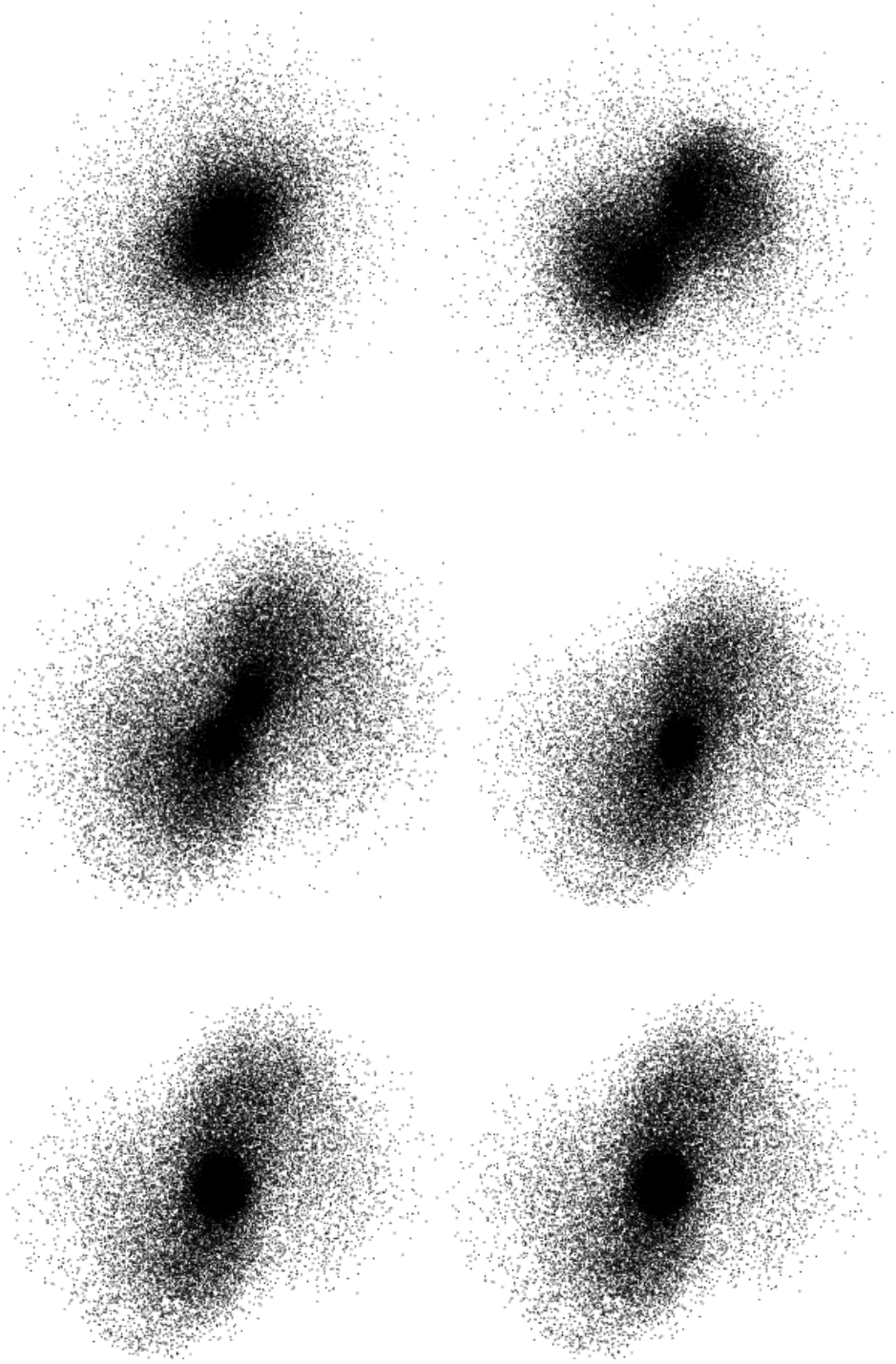
В рамките на всеки блок, частиците са подредени според техния вид, т.е. “gas” частиците са на първо място (тип 0), след това са частиците "halo" (тип 1), следвани от "disk" частиците (тип 2) и т.н.

Подробната последователност на частиците в блоковете се променя от изображение към изображение. Също така, дадена частица не винаги може да се съхранява във файл на моментно изображение с един и същи номер сред файловете, принадлежащи към един набор от файлове за моментно изображение. Това е така, защото частиците могат да се придвижват от един процесор на друг по време на паралелната симулация. С цел проследяване на частиците между различни изходи, се използват идентификационни номера (IDs) на частиците.

За да се позволи лесен достъп до данните във Fortran2, блоковете се съхраняват с използване на конвенцията "unformatted binary" от повечето реализации на езика Fortran. При този формат, данните са в блокове с определен размер, който представлява дължината на блок от данни в байтове. Тези блокове (които са две 4-байтови числа, съхраняващи стойностите преди и след блока) са полезни и в код на C за проверка на последователността на структурата на файла, както и да се осигури четете на точното място и бързо да се пропускат отделни блокове с помощта на информацията за дължината на блока, без да се налага да се прочитат последователно всички данни. Последното дава възможност да се четат ефективно само някои блокове, например само температури и плътности, като се пропускат координати и скорости.

На фиг. 9 е дадена визуализацията на моментните изображения – изходни данни от симулацията за сливане на две галактики, съдържащи съответно по 40000 и 20000 частици. Резултатите са записани в 8 моментни изображения.





Фиг. 9. Визуализация на моментните изображения – изходни данни от симулацията за сливане на две галактики, съдържащи съответно по 40000 и 20000 частици