Перевод компьютерных кодов для моделирования глюонной плазмы в сильном магнитном поле при конечной плотности на российскую ГРИД инфраструктуру РДИГ.

На данный момент задача вычисления характеристик кварк-глюонной плазмы требует больших затрат машинного времени и ресурсов памяти. К примеру, для получения зависимости величины кирального конденсата в SU(3) модели с улучшенным действием от значения внешнего магнитного поля (типичная задача) требуется, при грубой оценке, несколько десятков тысяч часов счёта без использования технологии параллельных вычислений. При задействовании кластеров или грид-систем с использованием порядка сотни процессоров это время можно уменьшить до нескольких суток, что является существенным улучшением. Таким образом, грид-технология является принципиальным решением проблемы ресурсов для программ, допускающих параллельное выполнение счета.

Коды, используемые для расчета характеристик глюонной плазмы включают в себя несколько основных блоков программ:

1. Программы для генерации конфигураций глюонных полей методом Монте-Карло. Написаны на языке C/C++.
2. Программы для расчета собственных значений и собственных функций оператора Дирака для различных значений внешнего магнитного поля и плотности (химического потенциала). Данные программы используют уже полученные конфигурации глюонных полей. Программы написаны на языке C/C++, частично на ASSEMBLER и FORTRAN.
3. Программы для расчета наблюдаемых типа электрической проводимости, кирального конденсата, магнитной восприимчивости, локального дипольного момента и т.д. Написаны на языке C/C++.
4. Математические библиотеки ARPACK, LAPACK, BLAS, необходимые для вычислений с использованием линейной алгеброй, в частности для операций с большими разреженными матрицами и реализации алгоритмов Arnoldi и Lanczos. Написаны на языке FORTRAN.
5. Управляющие скрипты, выполняющие компиляцию кода, формирующие входные файлы и аргументы командной строки перед запуском вычислений, а также скрипты, обрабатывающие результаты вычислений. Написаны на языках PERL, BASH и командном языке Gnuplot.
6. Скрипты и задания, предназначеные для работы с ГРИД (см. ниже). Написаны на языке BASH и JDL (gLite Job Description Language).

Программы из п.2 являются наиболее ресурсоемкими. На практике оказывается, что для обработки одной конфигурации глюонных полей размера 14^4 требуется около 1-3 суток и несколько сотен мегабайт оперативной памяти даже в случае нулевого химического потенциала. Для каждого значения магнитного поля выбирается около 30 конфигураций для достаточной точности. В силу независимости расчетов для каждой выбранной конфигурации их следует проводить параллельно, используя ресурсы РДИГ.

Использование уже имеющихся кодов (п.1-п.5) под ГРИД инфраструктурой без предварительной модификации невозможно в силу самого принципа устройства грид-систем, предполагающего работу программы под заранее неизвестной платформой и операционной системой.

В ходе такой модификации требуется выполнение следующих этапов:

1. Отладка работающего кода кода для конечной плотности для выделенного значения магнитного поля на локальном компьютере. Цель – заменить глобальные пути для используемых библиотек и исходных кодов на локальные, а также добавить в компилирующий скрипт автоматическое определение версий доступных компиляторов (Fortran, C/C++) и выборку наиболее подходящих из них. Также необходимо избавиться от привязки к определенной архитектуре, версии операционной системы, соответственно, выключить или изменить архитектурно-зависимые процедуры и процедуры, работающие только для компиляторов старых версий.
2. Отладка процесса постановки тестовых заданий, их отправки, приёма результата, отслеживания постановки в очередь, выполнения. Отладка соответствующих скриптов (п. 5 в списке программ).
3. Отладка кода на удаленном ГРИД-компьютере. Предполагается, что исходный код, необходимые библиотеки, а также тестовые решеточные конфигурации находятся в одном архиве, который пересылается на удаленную машину вместе со скриптом, его распаковывающим, компилирующим программу, выполняющим его и формирующим выходной итоговый файл. Цель – добиться правильности выполнения всех этапов работы, в частности, этапа сборки откомпилированных исходных файлов (написанных на разных языках программирования) в исполняемые файлы, а также этапа выполения программы, подключения динамических библиотек.
4. Регистрация решеточных конфигураций на элементах хранения ГРИД (storage elements). Цель – существенная экономия времени при отправке задания, а также возможность работы нескольких пользователей с одними конфигурациями.
5. Отладка кода на удаленном компьютере с использованием зарегистрированных конфигураций. Нахождение оптимального способа распределения ресурсов (параллельная обработка нескольких решеточных конфигураций, посылка нескольких заданий, выбор разных вычислительных элементов для задач различной сложности).

Компилирующий скрипт

В силу различия версий операционных систем, доступных компиляторов, путей к динамическим библиотекам и библиотекам общего пользования для различных компьютеров грид-системы необходимо компилировать и собирать весь набор исполняемых файлов перед их использованием на данной конкретной системе. Для этих целей используется скрипт, размещенный в Приложении 1. Смысл каждого этапа работы скрипта отражен в соответствующих комментариях.

Данный скрипт использует Makefile в основной директории, файл Makefile.var переменных для него, файл var_new переменных для gcc-компилятора версии выше 4-й, файл var_old для gcc версии ниже 4-й, файл 3dpart/Makefile для сборки математических fortran-библиотек. Смысловая часть файлов переменных, необходимая для понимания работы компилируещего скрипта:

```
---------var_old-----------------------------------------
...
DEBUG           =       0
TIMING          =       0
NCOLORS         =       3
EFIELD          =       1
MU              =       1
SINGLE          =       0
LAT_S           =       14
```

```
LAT_T           =       14

# Установки компилятора
CC                      =       ./gcc34
LD                      =       ./gcc34
FC                      =       g77

# Include dir
INC_DIR         =       ./include
# Source dir
SRC_DIR         =       ./src
# Where to put *.o files
OBJ_DIR         =       ./obj
#  Where to put binaries
BIN_DIR         =       ./bin
# Where to put libraries
LIB_DIR         =       ./lib
...
-----------------------------------------------------------

---------var_new-------------------------------------------
...
# Установки компилятора
CC                      =       gcc
LD                      =       gcc
FC                      =       gfortran
...
-----------------------------------------------------------
```

Скрипты для работы с вычислительными элементами ГРИД

Для работы с вычислительными элементами ГРИД была выбрана следующая стратегия: удаленную машину отправляется архив с исходным кодом программ (toverlap.tar), распаковывается при помощи управляющего скрипта(test.sh), далее производится выполнение программ, формируются выходные файлы с результатом работы. Для этого необходимо выполнить следующую последовательность команд:

Шаг 1. Инициализация прокси-сертификата:

```
voms-proxy-init --voms lattice.itep.ru
voms-proxy-info -all
```

Шаг 2. Проверка имеющихся вычислительных ресурсов:

```
voms-proxy-info -all
lcg-infosites --vo lattice.itep.ru all
glite-wms-job-list-match -a wn-info.jdl
```

Шаг 3. Если ресурсы имеются, поставить задачу в очередь:

```
glite-wms-job-submit -o job_id -a wn-info.jdl
```

Идентификатор задания будет записан в файл job_id.

Файл описания задачи:

```
---------wn-info.jdl-------------------------------------
VirtualOrganisation = "lattice.itep.ru";
Type = "Job";
# Выбор интерпретатора скрипта:
Executable = "/bin/sh";
# Управляющий скрипт:
Arguments = "test.sh";
# Файлы для потоков вывода и ошибок:
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err", "png.tar"};
# Файлы, отправляемые с локальной машины на удаленную
InputSandbox = {"toverlap.tar", "test.sh"};
```

```
# Выбор вычислительного элемента:
Requirements=RegExp("ce3.itep.ru",other.GlueCEUniqueID);
JobType = "Normal";
# Число попыток выполения задания:
RetryCount = 7;
-----------------------------------------------------------
```

Управляющий скрипт:

```
---------test.sh-----------------------------------------
tar -xvf toverlap.tar
cd toverlap
./compile.sh 14 14 1
cd bin
./run.pl
echo
cat out.dat
mv png.tar ../../
cd ../../
-----------------------------------------------------------
```

Запускающий скрипт run.pl (см Приложение 2) производит формирование аргументов командной строки, а также файла параметров для запуска вычислений. Кроме того, скрипт обрабатывает данные, выдаваемые программой для различных магнитных полей (в данном случае это значения корреляторов кварковых токов), аппроксимирует данные заданной функцией и строит графики для корреляторов и параметров аппроксимации в зависимости от магнитного поля (в данном случае это электрическая проводимость кварк-глюонной плазмы, а также масса ро-мезона)

Шаг 4. Проверка статуса выполнения задания (в очереди, выполняется, удалена и т.д.)

```
glite-wms-job-status -i job_id
# более высокий уровень подробности:
# glite-wms-job-status --verbosity 3 -i job_id
```

Шаг 5. Если сформирован выходной файл, переслать его на локальную машину:

```
glite-wms-job-output -i job_id
```

Шаг 6. (при необходимости) Удалить задачу:

```
glite-wms-job-cancel -i job_id
```

Шаг 7. (при необходимости) уничтожение прокси-сертификата:

```
voms-proxy-destroy
```

Скрипты для работы с файл-серверами ГРИД

Если имеются готовые данные о собственных значениях и функциях оператора Дирака, то их имеет смысл загрузить на доступный файл-сервер (storage element) ГРИД. Это необходимо сделать по следующим причинам:

1. Файл данных для каждой конфигурации с типичным размером (14^4, 16^4, 16^3x6 и т.д.) имеет размер от нескольких десятков до нескольких сотен мегабайт. Однократная загрузка данных на файл-сервер позволит существенно сэкономить время отправки каждой задачи.
2. Файл данных станет доступен другим пользователям ГРИД (в частности, членам виртуальной организации).
3. При улучшении метода генерации полей или обработки конфигураций прочие пользователи будут получать данные без необходимости копирования их на локальные машины.

Для загрузки файла «~/source» на файл-сервер виртуальной организации lattice.itep.ru так, чтобы он имел имя «~/destination» используется следующий скрипт (без переноса):

```
---------upload.sh source destination--------------------
lcg-cp -D srmv2 -b -v --vo lattice.itep.ru file:$HOME/$1
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$2
-----------------------------------------------------
```

Для копирования файла «~/source» с файл-сервера в текущую папку так, чтобы он имел имя «./destination», используется следующий скрипт (без переноса):

```
--------download.sh source destination-------------------
lcg-cp -D srmv2 -b -v --vo lattice.itep.ru
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$1 file:$PWD/$2
------------------------------------------------------------
```

Для удаления файла «~/source» с файл-сервера используется следующий скрипт (без переноса):

```
--------delete.sh source---------------------------------
lcg-del -D srmv2 -b -v -l --vo lattice.itep.ru
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$1
-----------------------------------------------------
```

Для оптимизации наиболее ресурсоемких расчетов, а именно расчетов собственных значений и собственных функций оператора Дирака, был разработан специальный скрипт использующий вычислительные элементов и файл-серверы ГРИД (см. Приложение 3).

Данный скрипт выполняет следующую последовательность действий:

1. Анализирует количество доступных конфигураций глюонных полей в локальной папке и копирует их на удаленный файл-сервер.
2. Для каждой из найденных конфигураций создает управляющий скрипт и файл входных параметров, которые будут отправлены на вычислительный элемент ГРИД. Управляющий скрипт включает в себя следующие основные этапы:
    a. Распаковка архива с иходным кодом программ.
    b. Запуск компилируещего скрипта.
    c. Копирование данных с файл-сервера.
    d. Выполнение собственно счета.
    e. Отправка конечных данных на файл-сервер.
3. Для каждой из конфигураций создается JDL-файл задания по уже указанному шаблону Отправляет полученное задание в очередь.

Таким образом, для каждой конфигурации выделяется отдельное задание, которое выполняется на доступных в данный момент ресурсах ГРИД инфраструктуры.

Приложение 1. Компилирующий скрипт.

```
---------compile.sh-----------------------------------
#!/bin/bash
# если существует gcc version 3.4, использовать его;
# в противном случае создать ссылку на текущую версию gcc
# в текущей папке
rm -f ./gcc34
if [ -e /usr/bin/gcc34 ]
then
        ln -s -f /usr/bin/gcc34 ./gcc34
else
```

```
        ln -s -f /usr/bin/gcc ./gcc34
        gcc -v
fi

# проверка, существует ли компилятор fortran77
if [ -e /usr/bin/g77 ]
then
        cp var_old Makefile.var
else
        cp var_new Makefile.var
fi

# следующая процедура может быть использована, если
# gcc-компилятор не сможет найти библиотеки общего
# пользования типа lapack в ./lib - папке.

#export TEMPR=$LD_RUN_PATH
#export LD_RUN_PATH=$LD_RUN_PATH:$PWD/lib/
#export TEMPL=$LD_LIBRARY_PATH
#export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/lib/


# компиляция математических fortran-библиотек
cd 3dpart
make clean
make all
cd ..


# Обработка аргументов, передаваемых скрипту из
# командной строки
# первый аргумент = число шагов по x,y,z
# второй аргумент = число шагов по времени t
# третий - точность вычислений (1=single, 0=double)
# без аргументов - параметры прошлого вызова скрипта

if [ "$#" != "0"  ]
then
        make LAT_S=$1 LAT_T=$2 SINGLE=$3 clean
        make LAT_S=$1 LAT_T=$2 SINGLE=$3 all
        make LAT_S=$1 LAT_T=$2 SINGLE=$3 mcurr
else
        make clean
        make all
        make mcurr
fi

#export LD_RUN_PATH=$TEMPR
#export LD_LIBRARY_PATH=$TEMPL
-----------------------------------------------------------
```

Приложение 2. Запускающий скрипт.

```
---------run.pl-------------------------------------------
#!/usr/bin/perl

$maxnfiles     = 100;

$beta          = 3.2810;
$spacing       = 0.1027;
$mass_phys     = 50.0;
$max_nz_evals  = 10;
$LS            = 14;
$LT            = 14;
$corrad        = 8;
@Hs           = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 20, 30);
$executable    = sprintf("./mcurr_%i%is", $LS, $LT);
$basedir       = sprintf("b%1.1f_s%i_t%i", $beta, $LS, $LT);

$NH = scalar(@Hs);
```

```
for($i=0; $i<$NH; $i++)
{
 $H = $Hs[$i];

 $paramname = sprintf("params.in");

 open(PARAMS, ">$paramname");
 printf PARAMS "%2.4f\n",  $beta;
 printf PARAMS "%2.4f\n",  $spacing;
 printf PARAMS "%i\n",     $H;
 printf PARAMS "%4.4lf\n", $mass_phys;
 printf PARAMS "%i\n",     $max_nz_evals;
 $outputfile = sprintf("b%2.4f_H%i_s%i_t%i_m%i_ev%i", $beta, $H, $LS, $LT,
int($mass_phys), $max_nz_evals);
 printf PARAMS "$outputfile\n";
 close(PARAMS);

 @files = glob("$basedir/H$H/ovd*.dat");

 $nfiles = scalar(@files);
 $nfiles = ($nfiles > $maxnfiles)? $maxnfiles : $nfiles;

 @files = @files[0..$nfiles];
 $arg = join(" ", @files);
 system("$executable $arg");

};

system("cat j_mu*.dat > out.dat");

open(PARAMS, ">all.plt");

printf PARAMS "set term png\n";
printf PARAMS "set fit errorvariables\n";

$NH = scalar(@Hs);

for($j=1; $j<5; $j++)
{

 printf PARAMS "set print \"fitparam$j.txt\"\n";

 for($i=0; $i<$NH; $i++)
 {
  $H = $Hs[$i];

  printf PARAMS "f%i_%i(x)=a%i_%i + c%i_%i*(exp(-m%i_%i*x)+exp(-m%i_%i*(%i-x)))\n",
$H, $j, $H, $j, $H, $j, $H, $j, $H, $j, $LT;
  printf PARAMS "fit f%i_%i(x) \"j_mu_nu_b%2.4f_H%i_s%i_t%i_m%i_ev%i.dat\" using 1:(-
\$%i):(\$%i) via a%i_%i,c%i_%i,m%i_%i\n", $H, $j, $beta, $H, $LS, $LT,
int($mass_phys), $max_nz_evals, 2*$j, 2*$j+1, $H, $j, $H, $j, $H, $j;
  printf PARAMS "print %i, a%i_%i, a%i_%i_err, m%i_%i, m%i_%i_err\n", $H, $H, $j, $H,
$j, $H, $j, $H, $j;

 };

};

printf PARAMS "set key outside\n";
printf PARAMS "set key width +1\n";
printf PARAMS "set xlabel \"H\"\n";
printf PARAMS "set ylabel \"Mass parameter\"\n";

printf PARAMS "set output \"mass_s%i_t%i.png\"\n", $LS, $LT;

printf PARAMS "plot \"fitparam1.txt\" using 1:4 title \"<j1 j1>\", ";
printf PARAMS "\"fitparam2.txt\" using 1:4 title \"<j2 j2>\", ";
printf PARAMS "\"fitparam3.txt\" using 1:4 title \"<j3 j3>\", ";
printf PARAMS "\"fitparam4.txt\" using 1:4 title \"<j0 j0>\"\n";

printf PARAMS "set ylabel \"additive constant\"\n";
```

```
printf PARAMS "set output \"aconst_s%i_t%i.png\"\n", $LS, $LT;

printf PARAMS "plot \"fitparam1.txt\" using 1:2 title \"<j1 j1>\", ";
printf PARAMS "\"fitparam2.txt\" using 1:2 title \"<j2 j2>\", ";
printf PARAMS "\"fitparam3.txt\" using 1:2 title \"<j3 j3>\", ";
printf PARAMS "\"fitparam4.txt\" using 1:2 title \"<j0 j0>\"\n";

printf PARAMS "set logscale y\n";
printf PARAMS "set xlabel \"tau\"\n";

for($j=1; $j<5; $j++)
{

       if ($j==4)
       {
              printf PARAMS "set output \"j0j0_s%i_t%i.png\"\n", $LS, $LT;
              printf PARAMS "set ylabel \"<j0 j0>\"\n";
       }
       else
       {
              printf PARAMS "set output \"j%ij%i_s%i_t%i.png\"\n", $j, $j, $LS, $LT;
              printf PARAMS "set ylabel \"<j%i j%i>\"\n", $j, $j;
       };

       printf PARAMS "plot ";

       for($i=0; $i<$NH; $i++)
       {
        $H = $Hs[$i];
        printf PARAMS "f%i_%i(x) with lines title \"fit H=$H\",
\"j_mu_nu_b%2.4f_H%i_s%i_t%i_m%i_ev%i.dat\" using 1:(-\$%i):%i with yerrorbars title
\"H=$H\" ", $H, $j, $beta, $H, $LS, $LT, int($mass_phys), $max_nz_evals, 2*$j, 2*$j+1;
        if ($i==($NH-1))
              {
                printf PARAMS "\n";
              }
               else
               {
                     printf PARAMS ", ";
               };
       };

}

close(PARAMS);


system('gnuplot all.plt');
system('rm all.plt');
system('tar -cf png.tar *.png');
-------------------------------------------------------------
```

Приложение 3. Расчет собственных значений и собственных функций оператора Дирака с использованием ресурсов ГРИД.

```
---------runtask.pl--------------------------------------
#!/usr/bin/perl

$LS    = 14;
$LT    = 14;
$H     = 1;
$E     = 0;
$m     = 0.00;
$beta  = 8.30;
$LOGNAME = sprintf("kalaydzhyan");

$basedir = sprintf("b%2.4f_s%i_t%i",$beta, $LS, $LT);
$datadir = sprintf("%s/H%i",$basedir, $H);
```

```
$wd      = sprintf("wd_su3_%i%id",  $LS, $LT);
$ovd     = sprintf("ovd_su3_%i%id", $LS, $LT);

print "\n\n\tDatadir: $datadir\n\n";

system("mkdir -p $datadir");
$configname = sprintf("%s/%ix%i*.dat", $basedir, $LT, $LS);

@configs = glob("$configname");
$nconfigs = scalar(@configs);

print("$nconfigs configurations found in $basedir\n");

for($i=0; $i<$nconfigs; $i++)
{
 $config      = $configs[$i];
 @configarray = split('/', "$config");
 $shortconfig = $configarray[scalar(@configarray)-1];
 $runfname    = sprintf("run%03id_H%i.sh", $i, $H);
 $jdlfname    = sprintf("task%03id_H%i.jdl", $i, $H);
 $infofname   = sprintf("lwd_%03i.info", $i);
 $wdfname     = sprintf("$datadir/wd_%03id_s%i_t%i_H%i.dat", $i, $LS, $LT, $H);
 $ovdfname    = sprintf("$datadir/ovd_%03id_s%i_t%i_H%i.dat", $i, $LS, $LT, $H);
 $ovdshortfname = sprintf("ovd_%03id_s%i_t%i_H%i.dat", $i, $LS, $LT, $H);

# system("lcg-cp -D srmv2 -b -v --vo lattice.itep.ru file:$config
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$shortconfig");

 open(INFOFILE,">$infofname");
 print INFOFILE "format    1\n";
 print INFOFILE "precision 4\n";
 print INFOFILE "lattice   $LS $LS $LS $LT\n";
 print INFOFILE "datafile  $basedir/$shortconfig\n";
 print INFOFILE "sun       3\n";
 print INFOFILE "endian    1\n";
 close(INFOFILE);

 open(RUNFILE,">$runfname");
 print RUNFILE "\#PBS -q long\n";
 print RUNFILE "\#PBS -l cput=199:59:59,mem=900mb\n\n";
 print RUNFILE "tar -xvf toverlap.tar\n";
 print RUNFILE "cd toverlap\n";
 print RUNFILE "./compile.sh $LS $LT 0\n";
 print RUNFILE "cd bin\n";
 print RUNFILE "mkdir $basedir\n";
 print RUNFILE "mkdir $datadir\n";
 print RUNFILE "mv ../../$infofname ./$basedir\n";
 print RUNFILE "lcg-cp -D srmv2 -b -v --vo lattice.itep.ru
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$shortconfig file:\$PWD/$basedir/$shortconfig\n";

 print RUNFILE "./$wd  -i $basedir/$infofname -o $wdfname -E $E -H $H -h -s -r 1.4 -a
-S 30 -V\n";

 print RUNFILE "./$ovd -i $basedir/$infofname -O $wdfname -o $ovdfname -E $E -H $H -r
1.4 -m $m -t 0 -S 20 -V -a\n";

 print RUNFILE "lcg-cp -D srmv2 -b -v --vo lattice.itep.ru file:\$PWD/$ovdfname
srm://selattice.itep.ru:8446/srm/managerv2?SFN=/dpm/itep.ru/home/lattice.itep.ru/$LOGN
AME/$ovdshortfname\n";

 close(RUNFILE);

 open(JDLFILE, ">$jdlfname");
 print JDLFILE "VirtualOrganisation = \"lattice.itep.ru\";\n";
 print JDLFILE "Executable = \"$runfname\";\n";
 print JDLFILE "Arguments = \"\";\n";
 print JDLFILE "StdOutput = \"std.out\";\n";
 print JDLFILE "StdError = \"std.err\";\n";
```

```
 print JDLFILE "OutputSandbox = \{\"std.out\",\"std.err\"\};\n";
 print JDLFILE "InputSandbox = \{\"$runfname\",\"$infofname\", \"toverlap.tar\"\};\n";
 print JDLFILE "Requirements=RegExp\(\"ce3.itep.ru\",other.GlueCEUniqueID);\n";
 print JDLFILE "JobType = \"Normal\";\n";
 print JDLFILE "RetryCount = 7;\n";
 close(JDLFILE);

 system("chmod 777 *.sh");
 system("glite-wms-job-submit -o job_id -a $jdlfname");
 system("rm -f $jdlfname");
 system("rm -f $runfname");
 system("rm -f $infofname");
};
-----------------------------------------------------------
```