

CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

JRA4 – Grid-enabled Treatment Planning System

Deliverable ID: D-JRA4-1b

Due date: January 31st, 2005

Actual delivery date: February 20, 2005

Lead contractor for this deliverable: IDRIS – CNRS, France

Project start date : May 1st, 2004

Duration: 5 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	X
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Content

Table of Content	2
1. Executive Summary	3
2. Introduction	4
3. Genomics environment, coupling with InfoBioGen	Erreur ! Signet non défini.
3.1 Purpose	Erreur ! Signet non défini.
3.2 Genomics database installation	Erreur ! Signet non défini.
3.3 BLAST installaion and optimization	Erreur ! Signet non défini.
3.4 Remote job soumission environment.....	Erreur ! Signet non défini.
4. Status of "Identification of new human mitochondrial proteins"	Erreur ! Signet non défini.
5. Status of "Large scale microbial genome re-annotation"	Erreur ! Signet non défini.
6. Annex.....	16

1. Executive Summary

This document is one of the two PM6 deliverables of the Joint Research Activity in Life Sciences. Initially scheduled for PM6, it is actually delivered at PM9. This application was initially because DEISA and EGEE decided to jointly deploy the Radiotherapy Treatment Planning. It describes the status of the two initial "early users" applications in the Genomics area that are planned in the work program. It also describes the activity being carried to deploy a software bio-informatics environment in the DEISA research infrastructure.

This document is publicly available.

2. Introduction

The area of Life Sciences is one of the most challenging ones in the context of high performance computing, because in most applications the important raw computing power or the data management facilities provided by the DEISA platforms has to be interfaced and integrated with external lightweight elements (Web interfaces, lightweight servers, etc) that are the ones that are accessed directly by the end users.

This is the reason why particular attention is being paid in this activity to portals that hide the complexity of the DEISA environment from end users. One of the first steps taken by IDRIS was to deploy a job submission environment that would allow computing centres specialized in Genomics to reroute some of their most demanding job requests to the DEISA platform, in a way transparent to the end users. This activity is not planned in the JRA4 work program, but it has been incorporated to it and reported here because of the basic role it plays in adapting the DEISA environment to the needs and requirements of the Life Sciences user communities.

3. The Radiation Therapy Planning computational environment

3.1 - Introduction

GATE (Geant4 application for Tomographic emission) [R14] is a C++ Monte-Carlo simulation platform based on the Geant4 toolkit. GATE was initially designed for nuclear medical imaging modalities such as Positron Emission Tomography and Single Photon Emission Computerized Tomography. Its functionalities, combined with its ease of use, make it particularly relevant for radiotherapy and brachytherapy treatment planning.

One of the key challenges related to the deployment of GATE in a grid environment is the perspective of offering services for radiotherapy treatment planning to medical physicists and physicians. Indeed, the accuracy of Monte Carlo (MC) dose computation is awesome, provided that the computing power is sufficient to allow for enough runs to reduce the statistical noise. The characteristics this type of computation allows efficient deployment on a loosely coupled grid of computing platforms, as we will discuss later on. MC dose computations could become standard for radiotherapy quality assurance (QA), planning, and plan optimization years before individual departments could afford a local investment that is capable to support MC. Requirements needed for such deployment include the existence of a service level agreement between the end users and the grid providers by which the grid level of performances in terms of security, stability and response time is guaranteed.

3.2 - Cancer treatment planning using GATE Monte-Carlo simulation

In this section, we describe a typical protocol for radiotherapy treatment. A patient comes to hospital in order to image the organ where a tumor has been previously located. Images are typically IRM and/or CT scans. Those images are produced in a DICOM format. After examination of the patient and analysis of medical images, the physician is able to make a diagnosis. He elaborates a prescription to kill the tumor. In our case, the medical cure is based on radiotherapy and brachytherapy. Both modalities require planning before treatment to compute exposure time of the different tissues to ionizing radiation.

Brachytherapy is the use of encapsulated radioactive sources to treat cancer. Radioactive sources are used to deposit therapeutic doses near tumours while preserving surrounding healthy tissues. In this case, Monte Carlo technique is used to optimize the dose calculations around the brachytherapy source.

Radiotherapy involves directing a beam of megavoltage x rays or electrons (occasionally protons) at a very complex object, the human body and especially a tumour. Currently, Monte Carlo simulation techniques are the most accurate method for dose calculation in radiotherapy, in particular when radiation is transported from one medium to another. The principle of a Monte Carlo simulation is to simulate the radiation transport knowing the probability distributions governing each interaction of particles in materials. Different possible trajectories or histories of a particle could be produced. Then, simulations store physical quantities of interest for a large number of histories to provide information on required quantities.

In the case of radiotherapy-brachytherapy applications, the goal is to calculate accurately the dose distribution in a located tumor. Most of the commercial systems, named TPS (Treatment Planning Systems), use an analytic calculation to determine these dose distributions, but errors can reach up to 10 to 20%. Such codes are very fast (execution time below one minute to give the dose distribution of a treatment), thus allowing their usage in medical centres.

Accurate calculations of Monte Carlo simulations involve a very high computing time compared to analytic calculations used in routine cancer treatment planning. Especially for radiotherapy treatments, Treatment Planning System (TPS) use approximations in the beam model and the dose calculation (e.g. the exclusion of electron transport) to speed up the computation. This may introduce significant uncertainties in the dose distributions in a patient, especially in the presence of heterogeneities such as the air-tissue, lung-tissue and tissue-bone interfaces. To comply with medical requirements, errors in the dose calculation should be kept below 2%. It is the reason why, for specific applications, the use of Monte Carlo simulations seems to be the best way to compute complex cancer treatment.

3.3 - Treatment planning computation in a grid environment

The computing time of a Monte Carlo simulation depends on different parameters: the number of particles generated during a simulation, the medium where particles interactions occur. Depending on the type of material filling the medium and the type of particles generated, the number of physical interactions can vary. So, there is a real

interest for parallel and distributed Monte Carlo simulations in order to provide very accurate studies in a reasonable amount of time. For this purpose, we are studying the deployment of GATE on PC farms and on supercomputers.

Distributed computation on PC farms

The Radiation treatment Planning application is very well adapted to the EGEE Grid. The GATE application relies on Monte Carlo methods to simulate the propagation of energetic particles in human tissues. Each particle trajectory is an independent event that is determined by some random initial condition and the nature of the tissues in which the particle propagate (an input for the computation). GATE is not a parallel application itself; it runs on only one processor. But the whole simulation can be parallelized with respect to the initial condition, and belongs to the well known “embarrassingly parallel” class. Different trajectories can be run on different processors, and the complete results can be collected at the end of all runs for statistical analysis. Each trajectory must of course be able to access the input data, but the grid portal developed by EGEE handles the dispatching of the input data to all the computing platforms involved in the simulation (as well as the recovery of the final results).

Computation on a supercomputer

Since the “embarrassingly parallel” nature of this application makes it very well adapted to the EGEE Grid environment, one may wonder about the interest of rerouting it to a supercomputer. GATE is a single processor application, and from the point of view of single processor performance, the DEISA platforms are probably more efficient than PC processors, but the difference is not overwhelming. Single processor performance alone does not justify the deployment on this application on the DEISA platform. This point was clearly recognized by the EGEE-DEISA technical teams involved in the collaboration.

The reason why this application has nevertheless been migrated to the DEISA platform is that the end user does not care about single processor performance; he cares about turnover times, namely, how much he waits to get the end result. Here, for some “mission critical” cases where turnover times must be as short as possible, the DEISA platform can make a difference. In the EGEE Grid turnover times are not fully guaranteed if the embarrassingly parallel application is run asynchronously in several scattered computing platforms. On the DEISA platform, a large number of processors can be synchronously allocated to an application. Moreover, DEISA will be able in due time to provide an “advanced reservation” service that will be able to pre-allocate substantial resources to GATE applications. The strategy is therefore meaningful, and one of the most interesting aspects of this collaboration will be to compare, once the application will be fully deployed, the relative merits and performance – in the sense of turnover times – of the two platforms, and to understand the application profiles that are best adapted to each one of them.

The IDRIS-CNRS supercomputing centre is the pilot DEISA grid site on which GATE platform has been installed. DEISA infrastructure fully exploits the network bandwidth and offers the possibility to redistribute the computational workload by migrating jobs across different supercomputers.

3.4 - Management of medical images

In order for computations to be executed on the grid environments, patient medical images have to be taken out of the hospital. This requires an interface between the end user (physician, medical physicist) inside the hospital and the grids. Images going out of the hospital must be anonymized in order to preserve patient privacy. As described on figure 1, the proposed interface is a web portal acting as an entry point to the grid where medical images can be uploaded by health practioners. These images will be anonymized before going out of the hospital on the portal.

In a first step, the portal will be in charge of preparing the job submission on both DEISA and EGEE infrastructures. In a second step, job management should be enhanced in order to choose the best grid infrastructure where to submit the jobs. Such decision can be based on the job parameters only and taken at the portal level. If the decision also takes into account the grid infrastructure status (network connection, workload, resource availability), EGEE and DEISA information systems need to be able to speak to each other.

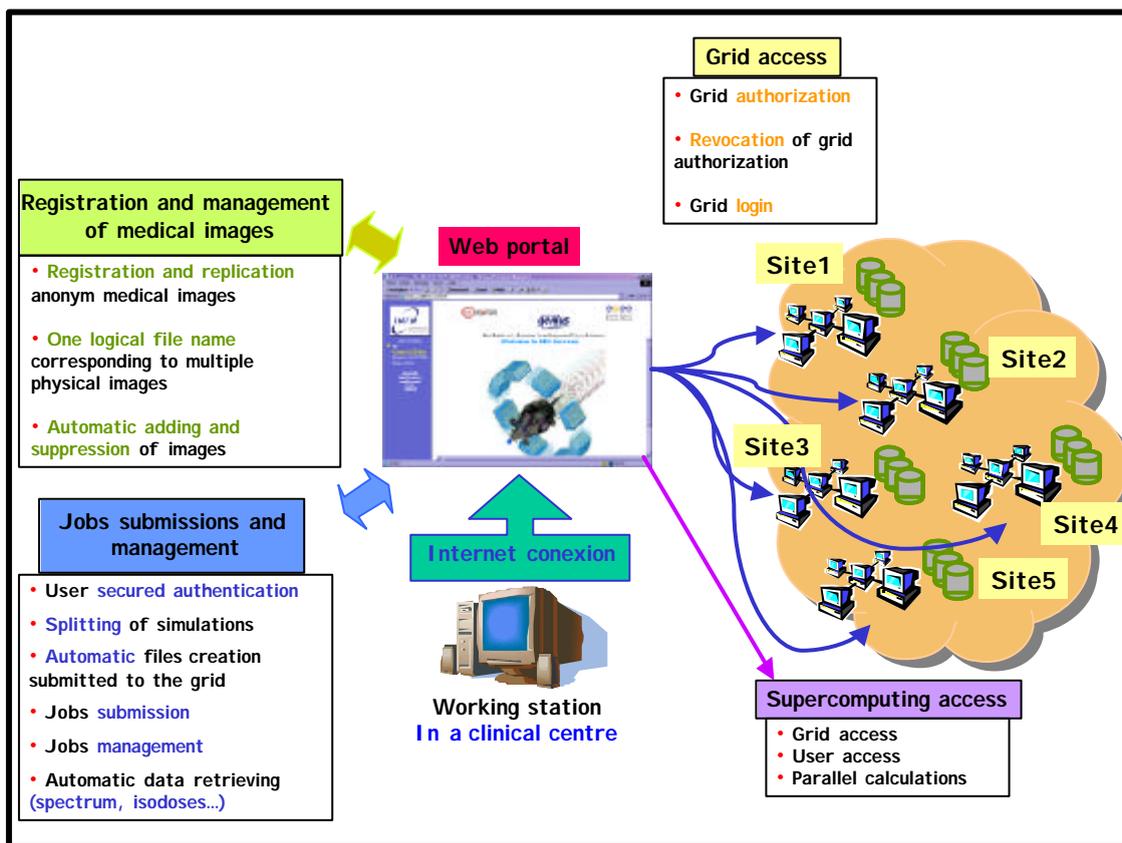


Figure 1 : schematic representation of the grid interface

The collaboration between EGEE and DEISA aims at defining a joint strategy to address the needs of medical physicists and physicians in clinical routine. The GATE platform is presently deployed on several EGEE nodes (CC-IN2P3, CINES Montpellier, LPC Clermont-Ferrand) and is under deployment on DEISA supercomputers at IDRIS [R7]. At a first step, the way to submit GATE simulations is under study. The parallelization technique has to be modified to take advantage of the supercomputing infrastructure in order to obtain consequent gain in terms of computing time and quick access to medical data. In a second step, the GENIUS grid portal (Grid Enabled web eNvironment for Independent User job Submission) is going to be customized to allow job submissions to both infrastructures. The portal functionalities will be enhanced in order to allow submission to either DEISA or EGEE depending on the jobs parameters (CPU time consumption, medical data computations...) and the grid sites' availability.

The use of a common application is an example of a pilot project where two infrastructure projects can combine their forces and provide better and more cost-effective service for a user community. This way a relatively large user community can gain an access to a unique, exceptionally efficient set of resources for their time-critical jobs. At the same time, the less urgent tasks can be handled by more affordable resources that are more readily available in a large-scale Data Grid infrastructure.

5. Annex

INSTALLATION OF THE *GATE* PACKAGE AT THE IDRIS IBM PLATFORM

DEISA TECHNICAL DOCUMENT

Isabelle DUPAYS, Anne FOUILLOUX, Denis RAUX
IDRIS - CNRS

1. Introduction

As a part of the collaboration between the DEISA and EGEE projects, joint applications are being deployed in the area of computationally intensive bio-medical applications.

The primary objective is to reduce the turnover time of Monte Carlo simulations in order to provide an efficient tool for specific cancer treatment requiring Monte Carlo accuracy. The simulations are parallelized on the Grid by splitting the initial conditions necessary to run Monte Carlo simulations. Each simulation is then computed using anonymous medical images of the tumor of the patient located on storage resources of the Grid.

A first step required the porting of the GATE application on the scalar supercomputers of the DEISA supercomputing platform. The porting was initially done on the IDRIS IBM-SP4 platform called "Zahir", with a total of 1024 processors. This paper describes the installation and validation procedures of the GATE application on this platform. The existence of a Common Production Environment in the DEISA AIX super-cluster implies that this application will run on any one of the nodes of this distributed supercomputer..

1.1 - GATE - Geant4 Application for Tomographic Emission

Radiotherapy and brachytherapy use ionizing radiations to treat cancer. Before each treatment, physicians and physicists plan the treatment using analytical planning systems and medical images data of the tumour. In order to treat patients with the best possible accuracy, Monte Carlo simulations are today the best tool to model and plan the tumour treatment for complex requirements. Relevant Monte Carlo simulations are however very time consuming, and this prevent hospitals and clinical centres to rely on them for current treatment planning.

GATE (Geant4 application for Tomographic emission) is a C++ software package based on the Monte Carlo Geant4 software. It has been typically designed to model nuclear medicine applications. Its functionalities combined to its ease of use make this platform also efficient for radiotherapy and brachytherapy treatment planning.

The origin of GATE can be traced back to a workshop organized in July 2001 in Paris attended by several research groups sharing a common interest in Monte Carlo simulations. Its focus was on the future of Monte Carlo simulations in nuclear medicine. The variety of drawbacks and limitations of the existent dedicated and general purpose codes were discussed. From these discussions, it became clear that it would be to everyone's interest to develop a simulation toolkit that would combine the best of both worlds.

The GATE source code resides in a CVS repository maintained by the OpenGATE Collaboration, available at the <http://wwwlphe.epfl.ch/~PET/research/gate/index.html> website. Documentation supports the simulation toolkit including installation and user guides, online source code documentation via [doxygen](#), and a list of frequently asked questions (FAQs). The code, documentation, and benchmarks are available from the restricted GATE [users](#) web site. Moreover, the e-mails exchanged on the user mailing list, and the FAQs are stored and directly accessible on these private web pages.

IDRIS is now a registered GATE user. We downloaded the 1.0.2 version of GATE and applied all necessary changes needed for our platform. To install and validate the GATE application, the following codes were also installed:

- ? CLHEP 1.8.2.0
- ? ROOT 3.10.02
- ? XPM 3.4
- ? GEANT4 5.2 patch 02

1.1.1 - CLHEP - A Class Library for High Energy Physics

CLHEP an acronym for a **C**lass **L**ibrary in **H**igh **E**nergy **P**hysics.

CLHEP is intended to be a set of HEP-specific foundation and utility classes such as random number generators, physics vectors, geometry and linear algebra. CLHEP is structured in a set of packages independent of any external package (interdependencies within CLHEP are allowed under certain conditions).

CLHEP source code resides in a CVS repository maintained by [Remi Mommsen](mailto:Remi.Mommsen@cern.ch) [<remigiusDOTmommsen@cern.ch>](mailto:Remi.Mommsen@cern.ch) and is available at <http://proj-clhep.web.cern.ch/proj-clhep/>.

1.1.2 - ROOT

ROOT tar files for the source, documentation and binaries are available at <http://root.cern.ch>. We installed only the ROOT library (not the root binary executable) because the "Zahir" platform is used only for scientific computation, not visualization. ROOT is only used to create and fill in the output file generated by gate simulations.

1.1.3 XPM

XPixmap (XPM) is available at <http://koala.ilog.fr/lehors/xpm.html>. XPixmap (XPM) consists of an ASCII image format and a C library. The format defines how to store color images (X Pixmap) in a portable and powerful way. The library provides a set of functions to store and retrieve images to and from XPM format data, being either files, buffers (files in memory), or data (included files).

While XPM is not an X Consortium standard, it is already a de facto standard. It is used in many commercial and non-commercial applications. Several vendors distribute the XPM library, as contributed software, on the platforms they sell. Moreover, the Common Desktop Environment specifies that icons must be stored either in XBM or XPM format. Finally, Motif 2.0 from OSF includes the XPM library, allowing XPM to be used in addition to XBM.

1.1.4 GEANT 4

Geant4 is a toolkit for the simulation of the passage of particles through matter. Its application areas include high energy physics and nuclear experiments, medical, accelerator and space physics studies.

Geant4 provides a complete set of tools for all the domains of detector simulation: Geometry, Tracking, Detector Response, Run, Event and Track management, Visualisation and User Interface. An abundant set of Physics Processes handle the diverse interactions of particles with matter across a wide energy range, as required by **Geant4** multi-disciplinary nature; for many physics processes a choice of different models is available.

In addition a large set of utilities, including a powerful set of random number generators, physics units and constants, Particle Data Group compliant Particle management, as well as interfaces to event generators and to object persistency solutions, complete the toolkit.

The **Geant4** source code is freely available at:

<http://wwwasd.web.cern.ch/wwwasd/geant4/geant4.html>,

accompanied by an Installation Guide and an extensive set of documentation.

2. GATE executable

The GATE software has not been ported on a number of platforms, but it has always been compiled and linked with the GNU C++ compiler (g++). The main goal of this porting is:

- Use of the IBM C++ compiler, xIC
- Adapting the code to the IBM-SP4 64 bits architecture

The GATE application was already ported at CINES on an IBM-SP4 (32 bit architecture). We have encountered many porting problems because most of these codes were developed before the ISO C++ 98 standard, and were later on validated only with the GNU C++ compiler.

There are a few major things the new ISO C++ 98 standard does:

- ? Clarify small differences between the various C++ compilers.
- ? Introduce namespaces and the std namespace (and the new include files)
- ? Introduce the new template "iostreams" library.
- ? Bring the STL (Standard Template Library) into part of the official Standard C++ Library.

The porting work required:

- ? Comparison tests with xIC version 6.0.0.0 and xIC version 7.0.0.0 on the IBM-SP4 zahir. These tests were performed with the collaboration of P.CORDE and showed that the compiler version 6.0.0.0 does not conform to the ISO C++ 98 standard. The new compiler xIC 7.0.0.0 was then used
- ? The source codes of GEANT4, ROOT and GATE were modified to conform the ISO C++ 98 standard. This step was necessary for the compilation of GATE and will facilitate all future porting activities.
- ? Tests with makesharelibC++ for the creation (and the call) of dynamic libraries: changes in Makefiles have been done to load dynamic libraries (for instance the following ROOT libraries: libfreetype.a or libCore.a) with an absolute path. We then install ROOT in a fixed location, not depending on ROOTSYS, a ROOT environment variable. This allows users to run ROOT without having to setup ROOTSYS and special PATH and LIBPATH variables. At the execution time, the Gate application is running on a local file system TMPDIR.
- ? Validation tests of the GATE application in collaboration with L.MAIGNE (LPC, Clermont-Ferrand).

2.1 Compilation/Link

2.1.1 CLHEP - 1.8.2.0

This version of CLHEP has been written to conform somewhat to the rules and recommendations described in the paper "Programming in C++ - Rules and Recommendations" written by Erik Nyquist and Mats Henricson (obtainable from freehep in the file <http://ftp.unicamp.br/pub/languages/c++/c++-rules.ps.gz>). Not all of the rules are followed however.

CLHEP is no longer supported on IBM AIX, but we didn't have any problems for the porting of CLHEP on Zahir. We reported all the changes made par A.EYNARD for the porting of CLHEP at CINES.

CLHEP was compiled with xIC 7.0.0.0 and at the end of the compilation process, a library is created (called libCLHEP-xIC_r.1.8.2.0.a).

2.1.2 ROOT - 3.10.02

ROOT didn't compile with the old version of xIC 6.0.0.0 because this version of xIC does not follow the ISO C++ 98 standard. We then compiled ROOT with xIC 7.0.0.0.

2.1.3 XPM 3.4

XPM, for compatibility reasons, was compiled with the new compiler version of xIC 7.0.0.0.

2.1.4 GEANT 4 - 5.2.p02

Two kinds of modifications were made:

- Update of the configuration files which define the compiler and link options.
- problems with C++ protected method which are not expanded in the include file
- C++ methods not declared (miss the corresponding include file) and used in C++ source files.

For the installation of GEANT4, the following environment variables were set:

```
export G4SYSTEM=AIX-xIC
export G4INSTALL=/usr/local/src/pub/GEANT4
export G4LIB=/usr/local/src/pub/GEANT4/lib
```

Note that GEANT4 was only tested on *32 bits architectures* with the Scientific Linux CERN 3 (SLC3) distribution (based on RedHat Linux Enterprise 3) and also with RedHat 7.3. Versions of Geant4 have also been compiled successfully on other Linux distributions, like Debian, Suse or more recent RedHat systems. GEANT4 has never been ported before on 64 bits architectures.

2.1.5 GATE - 1.0.2

We didn't use the source code downloaded at www-lphe.epfl.ch/~PET/research/gate/. We used instead the version already modified from GATE 1.0.2 by L.MAIGNE.

We performed the following modifications:

- Replace `<fstream.h>` with `<fstream>` to conform with the ISO C++ 98 standard.
- Change inline C++ methods when they were not expanded in the corresponding include file
- Cast with G4double dummy arguments of the mathematical function fabs

We defined the following environment variables:

```
export ROOTSYS=/usr/local/pub/root_v3.10.02
export G4SYSTEM=AIX-xlc
export G4INSTALL=/usr/local/pub/geant4
export G4LIB=/usr/local/pub/geant4/lib
export G4VERSION=5.2.p02
export CLHEP_BASE_DIR=/usr/local/pub/CLHEP
export CLHEP_INCLUDE_DIR=/usr/local/pub/CLHEP/include
export CLHEP_LIB_DIR=/usr/local/pub/CLHEP/lib
export CLHEP_LIB=CLHEP-xlc_r.1.8.2.0
export G4ANALYSIS_USE=1
export G4ANALYSIS_USE_FILE=1
export G4ANALYSIS_USE_ROOT=1
export G4ANALYSIS_USE_ROOT_PLOTTER=0
```

We then followed the installation guide (make and make global).

2.2 Validation

2.2.1 Validation of each component

Each installed software component was validated with the available benchmarks to check the installation. We obtained the expected results for all of them.

2.2.2 Performance validation of GATE

At this stage, the GATE application is running and gives the expected results. The next step is to compare the performance with the one obtained in other scalar platforms..

To be more realistic, L.MAIGNE provided us with an extensive test which ran on our IBM-SP4 as well as at LPC on a PC AMD athlon(tm) XP 1700+ running under Linux Red Hat 7.4. The generated ROOT file was then analyzed and showed that the simulation on both sites gave the expected results. Nevertheless, the initial performance analysis, given in the following table, gave on the IBM platform an execution time almost twice as big as the execution time on the PC platform:

LPC	IDRIS
22860 s	40980 s

This is not an unusual situation. C++ compilers are very sensitive to optimization options, and the options that work for one compiler do not necessarily work for another compiler. A large wealth of experience exists for the GNU compilers, but the

best way to optimize this code on the IBM xIC compiler is not yet clearly understood. This issue requires further analysis. We only have for the moment the following clues:

- I/O problems : the ROOT file which is filled in during the simulation could causes CPU overheads.
- Computational problems: we will recompile and link the GATE application with the « -p » option for a performance analysis. The profiler environment enables the identification of the computational bottlenecks of the application (the number of times that function was called, and the average number of milliseconds per call, for example). The full application can then be profiled with the gprof command.
- Use of totalview to debug the GATE application

3 Program changes

3.1 ROOT

We performed the following program changes to ROOT in order to install ROOT in a fixed location, not depending on ROOTSYS. This allows users to run ROOT without having to setup ROOTSYS and special PATH and LIBPATH variables.

We also add the "-bbigtoc" linker option to the compiler command line (enabling the "bigtoc" linker option) which allowed the build to continue, but this is the ld man page's comment:

bigtoc Generates extra code if the size of the table of contents (TOC) is greater than 64KB. Extra code is needed for every reference to a TOC symbol that cannot be addressed with a 16-bit offset. Because a program containing generated code may have poor performance, you should reduce the number of TOC entries needed by the program before using this option. The default is the nobigtoc option.

1. Changes in Makefile

```
RPATH := -L$(ROOTSYS)/$(LPATH)
```

instead of

```
RPATH := -L$(LPATH)
```

2. Changes in freetype/Module.mk

```
FREETYPELIB := $(ROOTSYS)/$(LPATH)/libfreetype.a
```

instead of

```
FREETYPE := $(LPCPATH)/libfreetype.a
```

3. Changes in build/unix/makelib.sh

```
if [ $PLATFORM = "aix5" ]; then
    makeshared="makeC++SharedLib"
fi
```

instead of

```
if [ $PLATFORM = "aix5" ]; then
    makeshared="/usr/vacpp/bin/makeC++SharedLib"
fi
```

and

```
echo $makeshared -bbigtoc -o $LIB -p 0 $OBS $EXTRA
$EXPLLNKCORE $makeshared -bbigtoc -o $LIB -p 0 $OBS $EXTRA
$EXPLLNKCORE
```

instead of

```
echo $makeshared -o $LIB -p 0 $OBS $EXTRA $EXPLLNKCORE
$makeshared -o $LIB -p 0 $OBS $EXTRA $EXPLLNKCORE
```

4. Changes in

```
unix/src/TUnixSystem.cxx
rpduils/src/ssh.cxx
rpduils/src/net.cxx
rpduils/src/netpar.cxx
```

```
#if defined(R__AIX) || (defined(R__FBSD) \
    && !defined(R__ALPHA)) || \
(defined(R__SUNGCC3) && !defined(__arch64__))
# define USE_SOCKETLEN_T
```

instead of

```
#if defined(R__AIX) || (defined(R__FBSD) \
    && !defined(R__ALPHA)) || \
(defined(R__SUNGCC3) && !defined(__arch64__))
# define USE_SIZE_T
```

3.2 CLHEP

A. EYNARD's changes made on CLHEP (when porting CLHEP on their IBM, CINES) reported on our 64 bits version.

3.2.1 Compilation

1. Changes in Matrix/GenMatrix.h (line 88)

```
pointer allocate(size_type n ) { if( n <= size ) return pool;
                               else return new T[n]; }
```

replaced with :

```
pointer allocate(size_type n, void * ) { if( n <= size )
                                         return pool;
                                         else return new T[n];
}
```

because the alloc function has 2 dummy arguments

2. Changes in Matrix/GenMatrix.h (line 100)

```
typedef std::vector<double,Alloc<double,25> > :: iterator
mIter;
typedef std::vector<double,Alloc<double,25> > :: const_iterator
mcIter;
```

replaced with :

```
typedef std::vector<double,Alloc<double,25> > mvect;
typedef mvect::iterator mIter;
typedef mvect::const_iterator mcIter;
```

because we obtained the following compilation error message when including the GenMatrix.h file in a source code:

```
xlc -c -DHAVE_CONFIG_H -I./.../.. DiagMatrix.cc
"./.../..CLHEP/Matrix/GenMatrix.h", line 100.16: 1540-0100 (S)
The class qualifier
"std::vector<double,HepGenMatrix::Alloc<double,25> >" contains
a circular reference back to "HepGenMatrix".
"/usr/include/gcc/darwin/3.3/c++/bits/stl_alloc.h", line
888.24: 1540-0700 (I) The previous message was produced while
processing "class HepGenMatrix::Alloc<double,25>". make[1]: ***
[DiagMatrix.o] Error 1
```

3. Changes in Matrix/SymMatrix.cc (line 67) et Matrix/Vector.cc (line 67)

```
#define SIMPLE_BOP(OPER) \
    register HepMatrix::mIter a=m.begin(); \
    register HepMatrix::mcIter b=m2.m.begin(); \
    register HepMatrix::mcIter e=m.begin()+num_size(); \
    for(;a<e; a++, b++) (*a) OPER (*b);
```

replaced with :

```
#define SIMPLE_BOP(OPER) \
    register HepMatrix::mIter a=m.begin(); \
    register HepMatrix::mcIter b=m2.m.begin(); \
    register HepMatrix::mIter e=m.begin()+num_size(); \
    for(;a<e; a++, b++) (*a) OPER (*b);
```

i.e. mcIter replaced with mIter.

4. Changes in HepPDT/DMFactory.icc (line 43)

```
if ( it == _makerfuncs.end() )
{
    std::auto_ptr<Product> result std::auto_ptr<Product>(0);
    return result;
}
```

replaced with :

```
if ( it == _makerfuncs.end() )
{
    std::auto_ptr<Product> result(0);
    return result;
}
```

to avoid the constructor and return variable alltogether.

3.2.2 Linking**1. Changes in HepPDT/DefaultConfig.hh(line 17 et 41)**

The following line was added at the beginning of this include file (line 17) :

```
#include "CLHEP/HepPDT/DMFactory.hh"
```

and line 41, we added the following line :

```
template class HepPDT::DMFactory<DefaultConfig>;
```

This modification is useful when linking gate with the CLHEP library.

3.3 GEANT4

3.3.1 Compilation

1. Changes in config/common.gmk (use of xlc instead of gcc)

We added (line 12):

```
G4CC := xlc -c
```

and line 104 :

```
$(G4TMPDIR)/%.d: src/%.cc
    @echo Making dependency for file $< ...
    @if [ ! -d $(G4TMPDIR) ] ; then mkdir -p
$(G4TMPDIR) ;fi
    @set -e; \
    $(G4CC) $(GPPFLAGS) $(CPPFLAGS) -w $< | \
    sed 's!$*\..o!$(G4TMPDIR)/& $@!' >$@; \

    [ -s $@ ] || rm -f $@
```

2. Changes in source/visualization/management/include/G4VViewer.hh (line 45)

We added:

```
friend class G4VSceneHandler;
```

in the *G4VViewer* class because *G4VSceneHandler* uses *ProcessView* is a protected method of *G4VViewer*. XIC returns the following compilation error message:

```
Compiling G4DAWNFILE.cc ...
source/visualization/management/include/G4VSceneHandler.hh",
line 288.15: 1540-0301 (S) The "protected" member
"G4VViewer::ProcessView()" cannot be accessed.
gmake[1]: *** [/usr/local/pub/geant4/tmp/AIX-
xlc/G4FR/G4DAWNFILE.o] Error 1
```

3. Changes in source/persistency/include/G4MCTgenParticle.hh et source/persistency/include/G4MCTEvent.hh

We had the following compilation error message:

```
Compiling G4MCTEvent.cc ...
"include /G4MCTGenParticle.hh", line 33.14: 1540-0139 (S) In
the context of the forward declaration, the name "Genvent" must
not be qualified. "include /G4MCTGenParticle.hh", line 34.14:
1540-0139 (S) In the context of the forward declaration, the
name "GenParticle" must not be qualified. "include
```

/G4MCTEvent.hh", line 41.14: 1540-0139 (S) In the context of the forward declaration, the name "Genvent" must not be qualified.
 "include /G4MCTEvent.hh", line 42.14: 1540-0139 (S) In the context of the forward declaration, the name "GenParticle" must not be qualified.

The following changes were made (given by the CCR Jussieu) :

- In source/persistency/include/G4MCTGenParticle.hh, comment lines

```
class HepMC::GenEvent;
class HepMC::GenParticle;
```

- insert in source/persistency/include/G4MCTEvent.hh of #ifndef MCT_GEN_PARTICLE_Hand #endif around

```
class HepMC::GenEvent;
class HepMC::GenParticle;
```

Changes in

source/visualization/management/include/G4VisManager.hh

We obtained the following compilation error message:

```
Compiling G4VGraphicsSystem.cc ...
"include/G4VisManager.hh" , line 348.22: 1540-0063 (S) The test
"*" is unexpected.
```

The following changes were made (given by the CCR Jussieu). Add the line:

```
#include "G4VisStateDependent.hh"
```

3.3.2 Linking

1. Changes in config/binmake.gmk (line 284)

```
@echo Linking $(G4TARGET) ...
@$(CXX) $(CXXFLAGS) $(CPPFLAGS) \
-o $(G4BINDIR)/$(G4TARGET) $(objects) $(LDFLAGS) \
$(LDLIBS)
```

replaced with :

```
# modif idris -bbigtoc

@$(CXX) -Wl,-brtl,-bloadmap:loadmap,-bbigtoc \
$(CXXFLAGS) $(CPPFLAGS) \
-o $(G4BINDIR)/$(G4TARGET) $(objects) $(LDFLAGS) \
$(LDLIBS)
```

This modification was proposed by A.EYNARD (CINES).

3.3.3 Execution

The following changes were made to avoid execution errors:

1. Changes in config/sys/AIX-xlC.gmk (from line 5 to 15)

```
ifeq ($(G4SYSTEM),AIX-xlC)
  CXX := xlC
  ifdef G4OPTIMISE
    CXXFLAGS := -O3 -qtwolink +-
  else
    ifdef G4DEBUG
      CXXFLAGS := -g -qdbxextra -qcheck=all -qfullpath -
qtwolink +-
      FCFLAGS := -g
      CCFLAGS := -g
    endif
  endif
endif
```

replaced with :

```
ifeq ($(G4SYSTEM),AIX-xlC)
  CXX := xlC
  CXXFLAGS := -qrtti=all
  ifdef G4OPTIMISE
    CXXFLAGS := -qrtti=all -O3 -qtwolink +-
  else
    ifdef G4DEBUG
      CXXFLAGS := -qrtti=all -g -qdbxextra -qcheck=all \
-qfullpath -qtwolink +-
      FCFLAGS := -g
      CCFLAGS := -g
    endif
  endif
endif
```

i.e. we added the « -qrtti=all » option to avoid a segmentation fault at execution time.

2. Changes in config/sys/AIX-xlC.gmk (from line 33 to 38)

```
define build-granular-shared-lib
@echo "Shared Libraries not supported on $(G4SYSTEM)
endif
define build-global-shared-lib
@echo "Shared Libraries not supported on $(G4SYSTEM)
endif
```

replaced with :

```
# Modif IDRIS :creation of dynamic libraries
define build-granular-shared-lib
```

```
@libdir=`(cd $(@D);/bin/pwd)`; \
cd $(G4TMPDIR); \ $(CXX) -G -o $$libdir/$(@F) $(INTYLIBS) *.o
endif
define build-global-shared-lib
@libdir=`(cd $(@D);/bin/pwd)`; \
cd $(G4TMP)/$(G4SYSTEM); \
$(CXX) -G -o $$libdir/$(@F) $(INTYLIBS) \
$(foreach dir,$(SUBLIBS),$(dir)/*.o);
endif
```

This modification was proposed by A.EYNARD (CINES).

3.4 - GATE

We started from the source codes provided by L.MAIGNE who already made a lot of changes to avoid execution errors.

3.4.1 Compilation

1. Changes in include/GateVObjectCreator.hh (line 219)

```
virtual inline void AttachCrystalSD() ;
///< Tell the creator that the logical volume should be attached
// to the phantom-SD
virtual inline void AttachPhantomSD() ;
```

replaced with :

```
// Modif IDRIS to remove inline because AttachCrystalSD is
// defined in another file
// 13/12/04 virtual inline void AttachCrystalSD() ;
virtual void AttachCrystalSD() ;
///< Tell the creator that the logical volume should be attached
// to the phantom-SD
// 13/12/04 virtual inline void AttachPhantomSD() ;
virtual void AttachPhantomSD() ;
```

*because when a function is declared **inline**, the function must be expanded at the calling block.*

2. Changes in GateSinogram.cc (from line 168 to 180)

```
if (fabs(crystallID - det1_c) < fabs(crystallID
- (det1_c +
(G4int)m_crystalNb))
    diff1 = crystallID - det1_c;
else
```

```

        diff1 = crystal1ID - (det1_c + m_crystalNb);
if (fabs(crystal2ID - det1_c) < fabs(crystal2ID
                                     - (det1_c +
(G4int)m_crystalNb)))
        diff2 = crystal2ID - det1_c;
else
        diff2 = crystal2ID - (det1_c + m_crystalNb);
if (fabs(diff1) < fabs(diff2)) sigma = crystal1ID - crystal2ID;

```

replaced with :

```

// Modif IDRIS fabs requires double arguments
if (fabs(G4double(crystal1ID - det1_c)) <
fabs(G4double(crystal1ID -
                                     (det1_c +
(G4int)m_crystalNb))))
        diff1 = crystal1ID - det1_c;
else
        diff1 = crystal1ID - (det1_c + m_crystalNb);

// Modif IDRIS fabs requires double arguments
if (fabs(G4double(crystal2ID - det1_c)) <
fabs(G4double(crystal2ID -
                                     (det1_c +
(G4int)m_crystalNb))))
        diff2 = crystal2ID - det1_c;
else
        diff2 = crystal2ID - (det1_c + m_crystalNb);

// Modif IDRIS fabs requires double arguments
if (fabs(G4double(diff1)) < fabs(G4double(diff2)))
        sigma = crystal1ID - crystal2ID;

```

to explicitly convert fabs arguments into double.

**Changes in sourceVoxelImageReader.cc (line 19) et
GateSourceVoxelTestReader.cc (line 18)**

```
#include "fstream.h"
```

replaced with :

```
// modif IDRIS <fstream> instead of "fstream.h"
#include <fstream>
```

to comply with the ISO C++ 98 standard.

3. Changes in GateToInterfile.cc (line 43)

We added the following include in order to define `BYTE_ORDER` (little or big-endian):

```
// Modif IDRIS added param.h for BYTE_ORDER
#include <sys/param.h>
```

3.4.2 Execution

1. Changes in include/GateToRoot.hh

```
//! Get the output file path
const char* GetFilePath() { return
(m_fileName+".root").c_str(); };
```

was replaced by:

```
// Modif IDRIS indicated by CINES
//! Get the output file path
// GetFilePath replace by the following code
const char* GetFilePath() { G4String s(m_fileName+".root");
char *res=new }
```

Without this modification, `GetFilePath` returns a null pointer. Indeed, the allocated memory at `(m_fileName+.root)` is immediately liberated and the chain does not exist anymore and cannot be used afterwards.