On the next pages you will find an example of exam subject (ISIMA/Master MSIR - HPC 2014-2015) - some questions may deal with the last lectures/labs

Master Research Students

- Come to the exam with your printed HPC survey - 3 pages according to a real scientific format: WSC or IEEE style.

- Send the PDF of your survey to this email address: David.Hill@univ-bpclermont.fr

  1st round of checks will be done with the compilatio software – good scientific journal do this kind of checks.

ISIMA - HIT & International curricula

- Come with a printed version of the report of your multi-agent simulation
- A demo will be done after the exam

Students following a biological cursus

- Send the PDF of your report to this email address:

David.Hill@univ-bpclermont.fr

HPC Exam – ISIMA F6/Master MSIR – February 2015

(1H30 - Documents allowed – no computers or cell phones)

Questions : Parallel stochastic simulation (4 pts)

Consider the random number generator given below. It is probably the simplest RNG to pass all of the TestU01 and dieHarder tests. The KISS generator (Keep It Simple Stupid) has been proposed by G. Marsaglia (a slightly older version is called KISS99 in a Pierre L'Ecuyer's paper). This remarkably short and fast generator combines 3 different simple generators and has a period of around 10^{37}.

```
/* Seed variables */
static unsigned int x = 123456789,y = 362436000,z = 521288629,c = 7654321;

unsigned int KISS()
{
   unsigned long long t, a = 698769069ULL;

   x = 69069 * x + 12345;
   y ^= (y << 13); y ^= (y >> 17); y ^= (y << 5);    /* y must never be set to zero! */
   t = a * z + c;
   c = (t >> 32);                                    /* Also avoid setting z = c = 0 */

   return x + y + (z = t);
}
```

1. Give additional comments for the code above to say if it is a True, a Quasi or Pseudo random number generator and to show how you understand the code inside the KISS function line by line (1 pt). Can you identify what are the kind of basic generators that compose this one? Precise their strength and weaknesses and if they are adapted to scientific and parallel computing (1 pt).
2. What kind of parallelization technique would you suggest to distribute random numbers with this generator for a parallel stochastic application and what are the other random number generators that you could advise (2 pts).

Questions (8 pts) – HPC concepts and techniques

1. What are the parallel file systems that you know and what can be their advantages? (2 pts)

2. What would be the advantages and drawbacks of an SMP machine compared to a computing cluster or grid? (2 pts)

3. If you need a workstation or computing node with peak performances what kind of machine and architecture would you propose? (2 pts)

4. What are the main benefits of hardware accelerators and what is their main drawback (2 pts)

Parallel computing and design (6 pts)

The Leibniz formula gives an approximation off PI.

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}.$$

Tough there are some better and faster approximations, this suite is faster than a Monte Carlo computing of PI to get significant decimals. This formula was formerly discovered in India around 1400 by Madhava, an Indian mathematician from the Kerala province. It corresponds to the Taylor limited development in 0 of the *arctan* function (evaluated in 1).

- Propose a sequential program in C implementing this suite (or in your preferred language) (1 pts)
- Consider you have 1000000 iterations and 1000 processors at your disposal – how could you distribute this sum to all the processors. Propose the necessary additional code to implement a parallel version of this sum (4 points).
- Discuss the potential problems you may encounter with this approach (1 points)

Understanding floating point arithmetic (2 pts)

Give the output of the following code and explain why you have selected this resulting output.

```c
int main()
{
    float a = 0.7;
    float b = 0.5;

    if (a < 0.7)
    {
       if (b < 0.5) printf("2 conditions are right");
       else         printf("only 1 is right");
    }
    else printf("0 conditions are right");
}
```