Feynman Loop Integral Computation on Hybrid Platforms
- Parallel computation of DCM -

primary author: E. de Doncker (Western Michigan Univ.)
co-authors: F. Yuasa (KEK) and R. Assaf (Western Michigan Univ.)
presented by F. Yuasa (KEK) in place of Elise

Plan of Talk

- Motivation
- Direct Computation Method (DCM)
- Parallel Computing of DCM
- Timing results
- Summary and future plans

Motivation

- High Precision theoretical calculations are required for LHC, ILC and other HEP experiments
- Higher order corrections require the evaluation of complicated Feynman loop diagrams

- Analytical approach
  - One-loop up to 4 legs + reduction for more legs
  - Difficulty: more loops, more legs, various physical parameters
- Numerical approach
  - We propose DCM
    - A fully numerical method
    - Up to two-loop with 4 legs with masses
    - Difficulty: numerical cancellation, long computation time

Feynman loop integrals for L-loops with N internal lines

Scalar integral

$$(-1)^N \left(\frac{1}{4\pi}\right)^{nL/2} \Gamma(N - nL/2) \int_0^1 \prod_{i=1}^{N} dx_i \delta(1 - x_1 \cdots - x_N) \frac{C^{N-n(L+1)/2}}{(D - i\varepsilon C)^{N-nL/2}}$$

D and C functions are polynomials of Feynman parameters $\{x_i\}$

Direct Computation Method (DCM) Ref. CPC 159 (2004) 145

DCM is a fully numerical method
Combination of numerical multivariate integration
and numerical extrapolation($\varepsilon \longrightarrow 0$)

Program flow of DCM

1st step

Let ε be finite as $\varepsilon_l = \dfrac{\varepsilon_0}{(A_c)^l}, \quad A_c > 1$

with l=0,1,2,…

ε_0 and a constant A_c are positive numbers

ε_0 and A_c are chosen empirically

2nd step

Evaluate the integral $I(\varepsilon_l)$ numerically and get the sequence of $I(\varepsilon_l)$ with l=0,1,2,…

We are using DQAGE routine for multivariate integration
This step is time consuming

(DQAGE : www.**netlib**.org/**quadpack**/)

3rd step

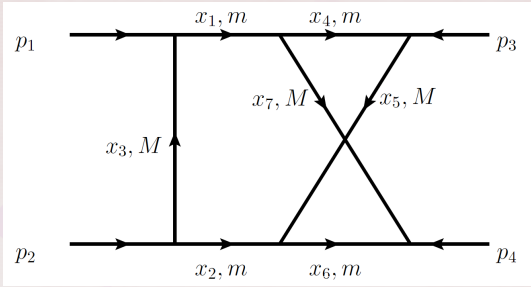
Extrapolate the sequence $I(\varepsilon_l)$ to the limit ($\varepsilon \to 0$) and determine I

We are using Wynn's epsilon algorithm
Computation time is negligible

Example of computation time of DCM two-loop non-planar box with masses in physical region

(numerical results were presented in ACAT2011)

Ref. CPC 183 (2012)2136



Real Part m=50 GeV, M = 90 GeV, t = -100^2 GeV^2

fs= s/m^2	Computation time	key	Limit
6.0	16 hours	2	10, 20, 10, 10, 10, 10
7.0	2 days	2	10, 20, 10, 10, 10, 10
10.0	1 week	2	10, 10, 10, 10, 10, 10

For f_s = -1, computation time is ~24 sec.

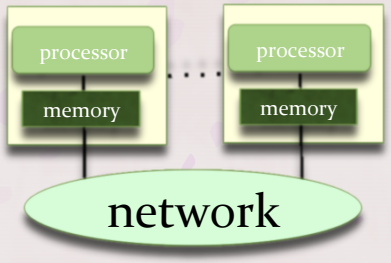
by single CPU: Intel(R) Xeon(R) CPU X5460 @ 3.16GHz

Integration parameter
Key : Gauss-Kronrod rule, 10 - 21 points when key = 2
Limit: an upperbound on the number of subintervals

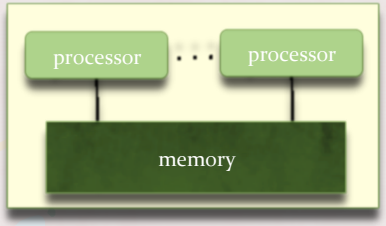
Parallel computing is required for the numerical integration

Parallel computing

- Parallel programming model
  - Distributed memory
    - MPI (de facto standard)
  - Shared memory
    - OpenMP (de facto standard)
    - Available with many compilers
      - “f90”, “gfortan” , “ifort ”, “pgi” …
- Acceleration of computing using GPUs



Distributed memory



Shared memory

Our approach

Parallelization of DQAGE routine for a numerical integration using OpenMP

Parallel numerical iterated integration

- DQAGE: a 1D adaptive integration code
  - The integration is performed with the (7, 15)- or the (10, 21) –points Gauss-Kronrod pairs.

$$\int_a^b dx_j \; F(c_1, \ldots, c_{j-1}, x_j) \approx \sum_{k=1}^{K} w_k F(c_1, \ldots, c_{j-1}, x^{(k)})$$

w_k : weights
$X^{(k)}$: abscissae

This can be evaluated in parallel

- The current implementation allows for a nested parallelization in the outer and the next to outer level.

Nested parallelism

- For example, DQAGE with 15-point (21-point) Gauss-Kronrod pair: best improvement around 15 (21) threads; then tends to level off.
- For example, performance with 15-point rule is excellent on 16-core node. What about 32-core node?
- Additional performance can be obtained by nested threading. For example,

  - 15 threads assigned to the rule eval. on the outer (x_1 level);

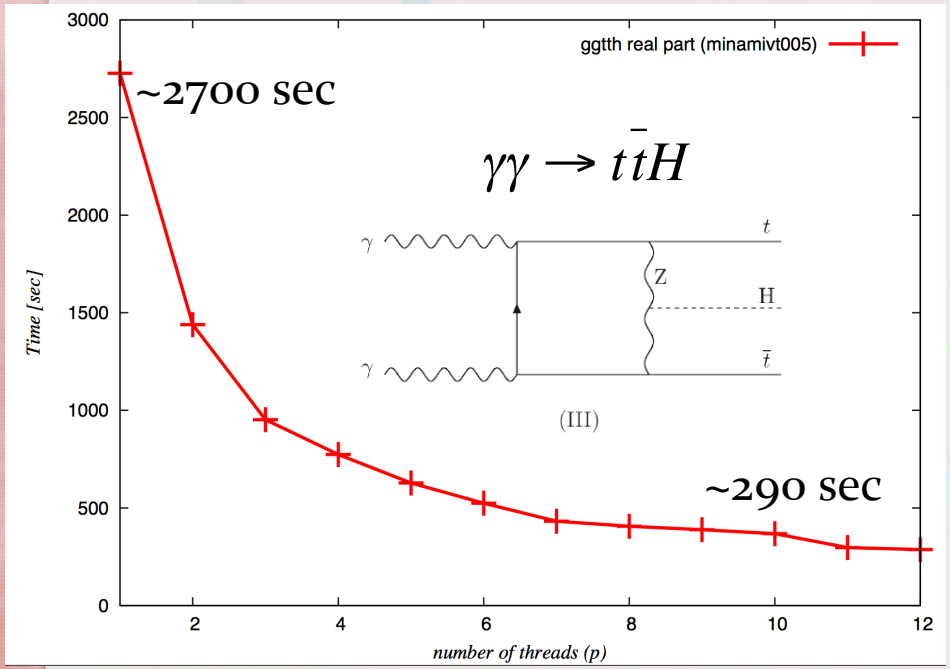
  - each function eval. on x_1 level is an integral on x_2 level;

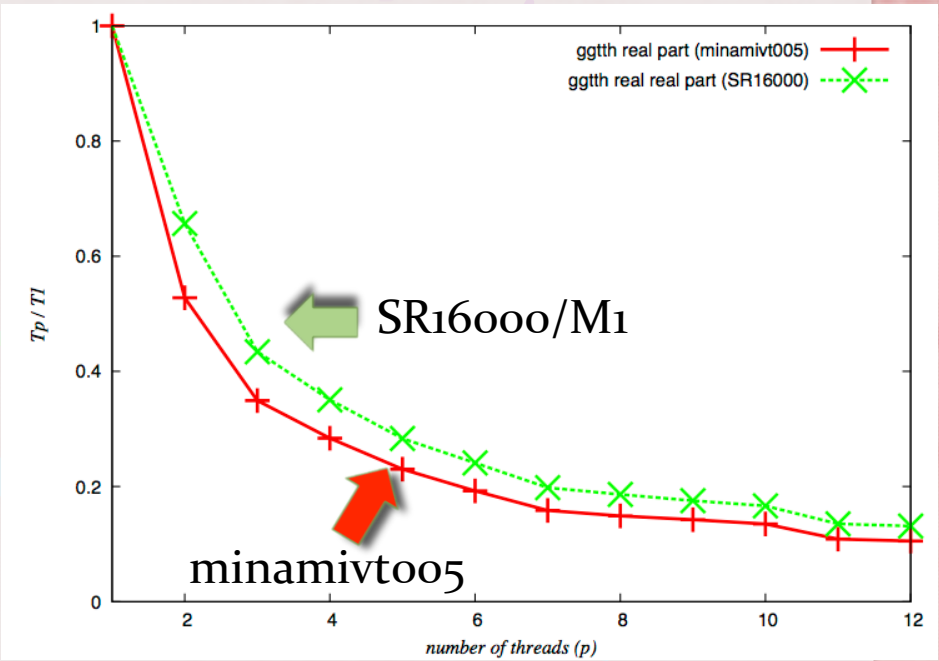
  - for each thread on x_1 level, new threads spawned on x_2 level.

Timing results on computers below

hosts	CPU	# of cores	compiler
Minamivt005 (KEK)	Xeon X-5680, 3.3 GHz	6 cores/ node	ifort -openmp
SR16000 / M1 (KEK)	Power7, 3.83 GHz	32 cores/ node	f90 -omp
Intel cluster (WMU)	Xeon E5-2670, 2.6 GHz	16 cores/ node	gfortran -fopenmp

1st example
one-loop integral with 5 legs





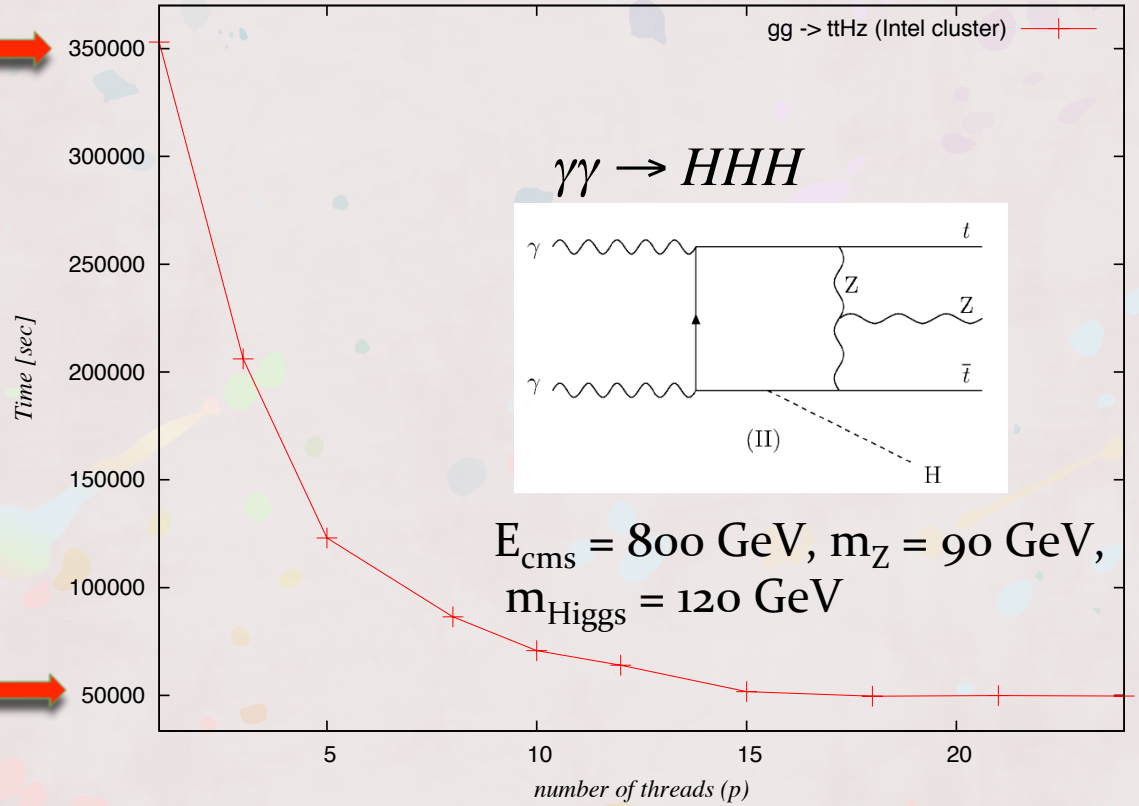
Same physical parameters as those in
Binoth, Heinrich, Kauer Nucl. Phys. B654(2003)277.

E_{cms} = 800 GeV, m_Z = 90 GeV, m_{top} = 175 GeV, m_{Higgs} = 120 GeV

2nd example
one-loop integral with 6 legs

4 days

13 hours



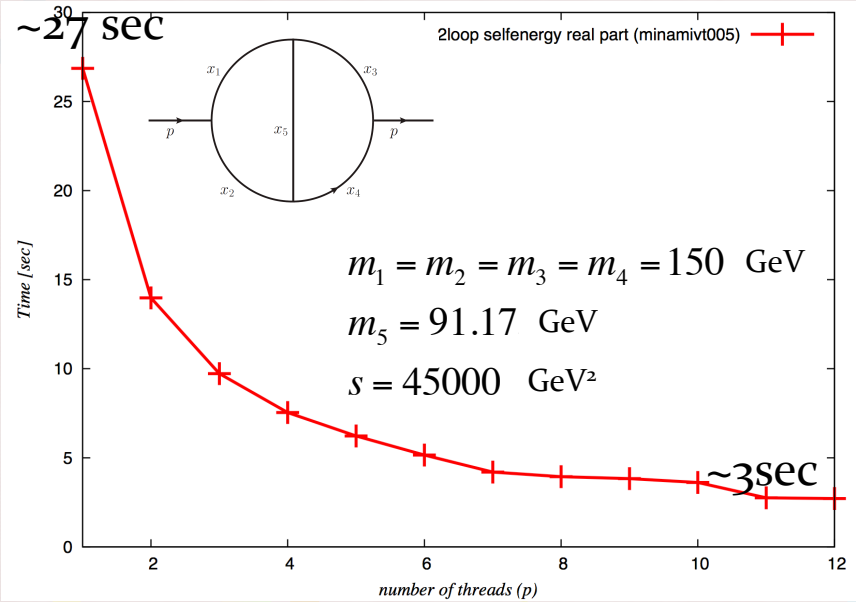
$\gamma\gamma \rightarrow HHH$

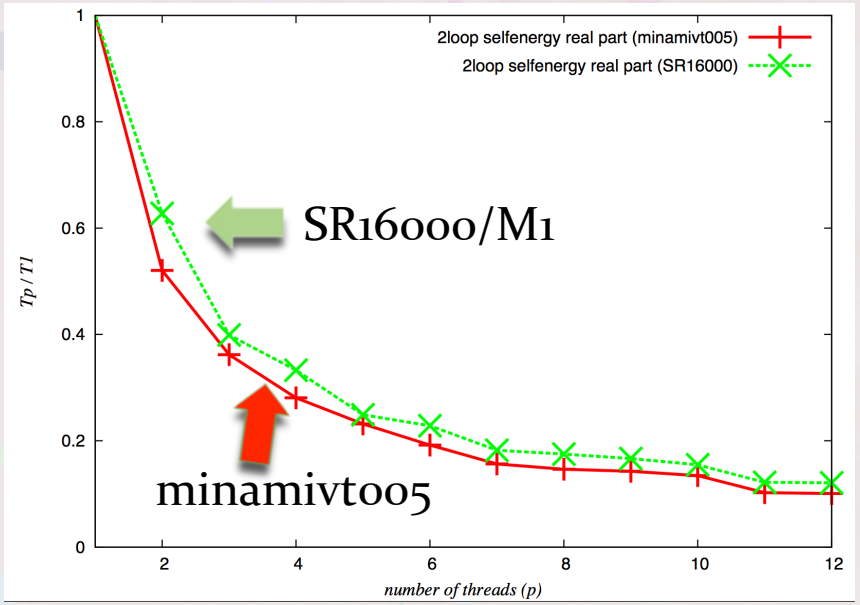
E_{cms} = 800 GeV, m_Z = 90 GeV,
m_{Higgs} = 120 GeV

Timing plot on WMU cluster (Xeon E5-2670, 16 cores /node)

Same physical parameters as those in
Binoth, Heinrich, Kauer Nucl. Phys. B654(2003)277.

3rd example
two-loop self-energy integral





$$I = \int_0^1 dx_1 dx_2 dx_3 dx_4 dx_5 \delta(1 - \sum_{i=1}^{5} x_i) \frac{1}{CD}$$

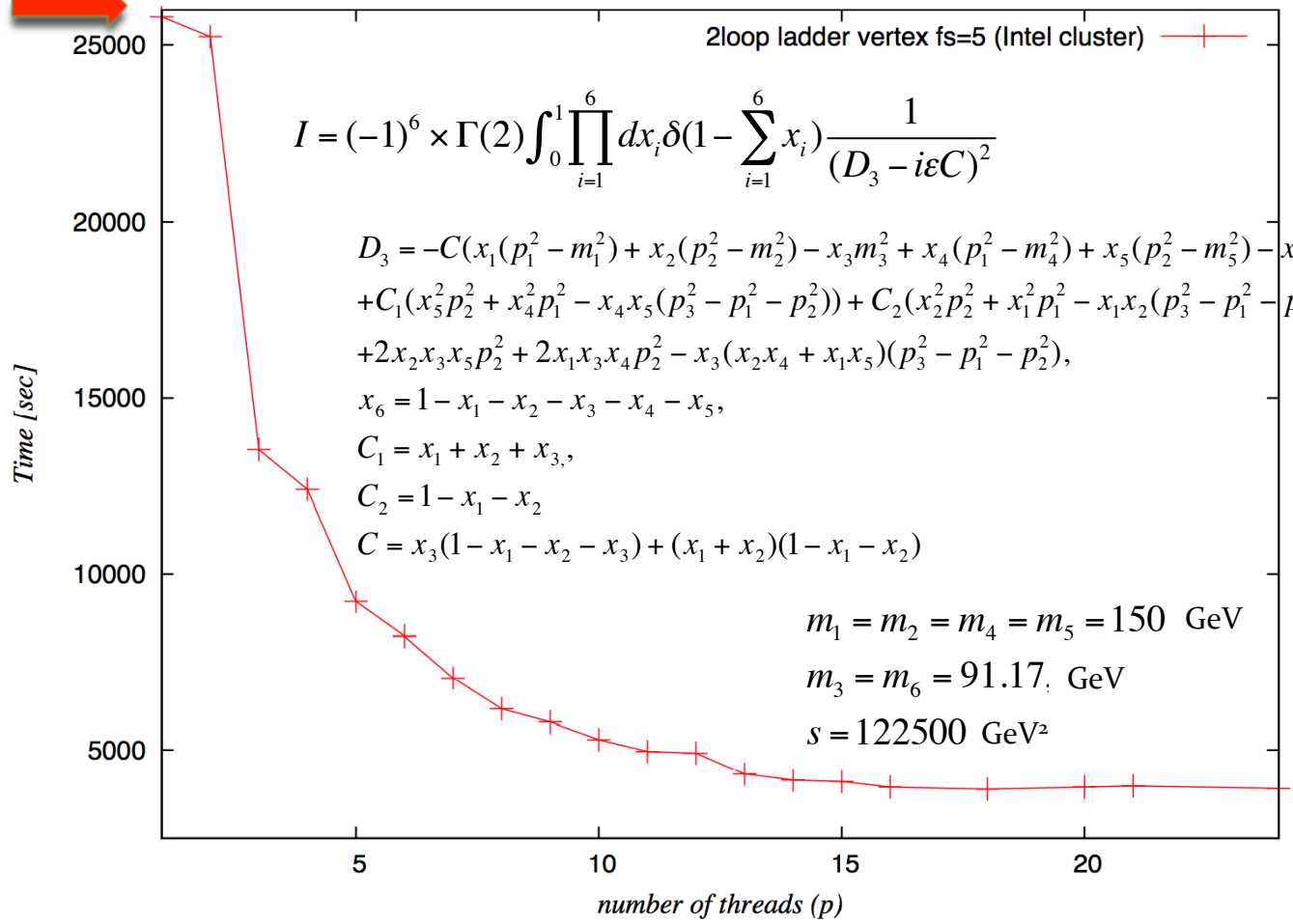
$$D = -s(x_5(x_1 + x_3)(x_2 + x_4) + (x_1 + x_2)x_3 x_4 + (x_3 + x_4)x_1 x_2) + C\tilde{M}^2,$$

$$C = (x_1 + x_2 + x_3 + x_4)x_5 + (x_1 + x_2)(x_3 + x_4),$$

$$\tilde{M}^2 = \sum_{i=1}^{5} x_i m_i^2.$$

4th example
two-loop vertex integral

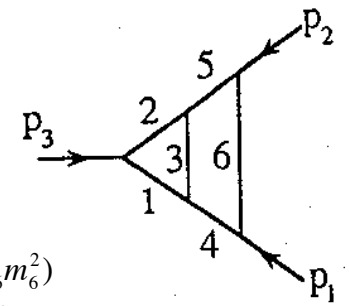
7 hours



$$I = (-1)^6 \times \Gamma(2) \int_0^1 \prod_{i=1}^{6} dx_i \delta(1 - \sum_{i=1}^{6} x_i) \frac{1}{(D_3 - i\varepsilon C)^2}$$

$$D_3 = -C(x_1(p_1^2 - m_1^2) + x_2(p_2^2 - m_2^2) - x_3 m_3^2 + x_4(p_1^2 - m_4^2) + x_5(p_2^2 - m_5^2) - x_6 m_6^2)$$
$$+C_1(x_5^2 p_2^2 + x_4^2 p_1^2 - x_4 x_5(p_3^2 - p_1^2 - p_2^2)) + C_2(x_2^2 p_2^2 + x_1^2 p_1^2 - x_1 x_2(p_3^2 - p_1^2 - p_2^2))$$
$$+2x_2 x_3 x_5 p_2^2 + 2x_1 x_3 x_4 p_2^2 - x_3(x_2 x_4 + x_1 x_5)(p_3^2 - p_1^2 - p_2^2),$$
$$x_6 = 1 - x_1 - x_2 - x_3 - x_4 - x_5,$$
$$C_1 = x_1 + x_2 + x_3,$$
$$C_2 = 1 - x_1 - x_2$$
$$C = x_3(1 - x_1 - x_2 - x_3) + (x_1 + x_2)(1 - x_1 - x_2)$$

$$m_1 = m_2 = m_4 = m_5 = 150 \text{ GeV}$$
$$m_3 = m_6 = 91.17 \text{ GeV}$$
$$s = 122500 \text{ GeV}^2$$



1 hour plus
a few min.

Timing plot on WMU cluster (Xeon E5-2670, 16 cores /node)

With nested threading on 32-core node

- Previous plot was obtained on 16-core nodes.
- Timings on 32-core node with 15-point rules, (j x k: j threads on x_1 level and k threads on x_2 level) example times:
  - 15 x 1: 6632 sec
  - 30 x 1: 5909 sec
  - 15 x 15: 3630 sec (gives best results)

Summary and future plans

- DCM (Direct Computation Method) is a fully numerical method for evaluating Feynman loop integrals. It is available for two-loop integrals with masses.
- It is clearly shown that the computation time of loop integrals by DCM can be reduced using OpenMP with parallelized DQAGE routine in QUADPACK.
- Future plans
  - We will do the timing evaluation for more computational intensive loop integrals such as two-loop box integral.
  - The current parallel DQAGE allows for a nested parallelization in the outer and the next to the outer level of the multivariate integration. This can be extended to more levels.

Summary and future plans (cont'd)

- Hybrid Platforms
  - **Use of MPI**
    - As in ParInt (http://www.cs.wmich.edu/parint/) , with region partitioning and region evaluations distributed over the nodes (+ load balancing), and multi-threading on the nodes.
    - For sets of integrals, e.g., as resulting from reductions: nodes obtain new problem specifications from task pool or task server upon request; multi-threading within each node. For example, hexagon reduction into 6 pentagons, or distribution on box level (individual problems benefit from multi-threading).
  - **Use of GPUs and other accelerators (and MPI/GPUs)**
    - GPUs: for 'regular' rules (grid/lattice-QMC rules and types of MC)
    - Intel Xeon Phi 5110P coprocessor with 60 cores and 8GB memorys

Thank you !