Stochastic optimization of GeantV code by use of genetic algorithms

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 042026 doi:10.1088/1742-6596/898/4/042026

Stochastic optimization of GeantV code by use of genetic algorithms

G.Amadio[1], J.Apostolakis[2], M.Bandieramonte[2], S.P.Behera[3], R.Brun[2], P.Canal[4], F.Carminati[2], G.Cosmo[2], L.Duhem[5], D.Elvira[4], G.Folger[2], A.Gheata[2,5], M.Gheata[2,6], I.Goulas[2], F.Hariri[2], S.Y.Jun[4], D.Konstantinov[2], H.Kumawat[3], V.Ivantchenko[2], G.Lima[4], T.Nikitina[2], M.Novak[2], W.Pokorski[2], A.Ribon[2], R.Seghal[3], O.Shadura[2], S.Vallecorsa[2,5] and S.Wenzel[2]

[1] Parallel Computing Center at Sao Paulo State University (UNESP), Sao Paulo, Brazil
[2] CERN, Route de Meyrin, Meyrin, Switzerland
[3] Bhabha Atomic Research Centre (BARC), Mumbai, India
[4] Fermilab, MS234, P.O. Box 500, Batavia, IL, 60510, USA
[5] Intel Corporation, Santa Clara, CA, 95052, USA
[6] Institute of Space Sciences, Bucharest-Magurele, Romania

E-mail: oksana.shadura@cern.ch

Abstract. GeantV is a complex system based on the interaction of different modules needed for detector simulation, which include transport of particles in fields, physics models simulating their interactions with matter and a geometrical modeler library for describing the detector and locating the particles and computing the path length to the current volume boundary. The GeantV project is recasting the classical simulation approach to get maximum benefit from SIMD/MIMD computational architectures and highly massive parallel systems. This involves finding the appropriate balance between several aspects influencing computational performance (floating-point performance, usage of off-chip memory bandwidth, specification of cache hierarchy, etc.) and handling a large number of program parameters that have to be optimized to achieve the best simulation throughput. This optimization task can be treated as a black-box optimization problem, which requires searching the optimum set of parameters using only point-wise function evaluations. The goal of this study is to provide a mechanism for optimizing complex systems (high energy physics particle transport simulations) with the help of genetic algorithms and evolution strategies as tuning procedures for massive parallel simulations. One of the described approaches is based on introducing a specific multivariate analysis operator that could be used in case of resource expensive or time consuming evaluations of fitness functions, in order to speed-up the convergence of the black-box optimization problem.

1. Introduction

The work is focused on the research of stochastic optimization algorithms and unsupervised machine learning methods involving multivariate analysis for tuning High Energy Physics (HEP) Monte-Carlo simulations with GeantV [1]. Simulation is the HEP computing activity accounting for a sizable fraction of the WLCG cycles.

Stochastic optimization refers to the minimization or maximization of a function in the presence of randomness in the optimization process. Assuming the use of a efficient set of genetic

algorithms suitable for "black-box" multi-objective problem (MOP) with computationally-expensive evaluations of fitness function (constrained or unconstrained Non-Dominant Sorting Genetic Algorithms (NSGA-II [2] or NSGA-III [3] algorithms)), the main problem in the schema of algorithms is a lack of additional checkers or operators that allow a faster convergence to the set of global optimum points.

For GeantV simulations, we proposed a set of parameters that are considered as the most important for tuning the applications performance. These are explicitly the run time, memory consumption and other model-specific tuning knobs. Adding specific operators to the genetic algorithm can be regarded as a noise reduction factor for faster approximation and convergence to the true Pareto front. This front consists of ideal individuals selected based on the set of parameters, while we can apply orthogonal transformation and emphasize variation to discover strong patterns in data.

2. GeantV concurrency model

During the last decade, Geant4 [4] has been the application of choice for full detector simulation. Its complex design having deep class structure hierarchies and calling stacks makes however hard to exploit at best modern architectures, generating low values for the instructions retired per cycle (IPC) and important cache misses.

The GeantV project started in 2013, pursuing a thorough R&D in several areas, including instruction level parallelism and aiming at increasing the throughput by enabling SIMD in the code and improving locality to avoid unnecessary load/stores and instruction fetches. This allowed GeantV-based applications to execute efficiently on many-core vector architectures such as Intel KNL and to optimize the usage of the system caches. The improved event throughput is expected to partially compensate the increasing need for simulated data samples for the High-Luminosity LHC operating regime. GeantV is getting significant benefits from the use of optimized components, such as a new geometry modeler library that provides several novel features including vectorization and transparent access to multiple computing architectures. The VecGeom library [5] is stateless and able to handle vectors of tracks in the computing-intensive parts, being adapted to heavy multi-threaded workloads. One of the advantages of VecGeom compared to similar libraries is its high portability to accelerators (GPU, KNL, etc.), taking advantage of their specificity.

Discussing GeantV model optimization, we can quote the "eight dimensions of parallelism" [6], where fine-grain parallelism is the only method allowing sustainable gains in both throughput and in time-to-solution. Other methods have their own merits: hardware threading has a very little gain to be expected and no action to be taken, multi-core parallelism allow reducing the memory footprint and time-to-solution but not the throughput, while multi-node (HPC) level gives the possibility of running many parallel jobs. These different options are all accessible to GeantV, increasing the dimensionality of the space of parameters to be optimized. Our idea is to try to optimize the simulation using a specific machine learning (ML) toolkit by increasing the dimensionality coverage of the processing jobs.

The main functionality of the GeantV core is to provide data structures and concurrent services to dispatch and control the concurrent work flow, enforce work coherence by packing particles into vectors, and push these vectors down to the SIMD-optimised computing intensive geometry and physics algorithms. A run manager is the module allowing to implement a given run configuration and simulation parameters. The run manager controls also the event loop and the workflow, injection of generated events in the system, concurrency parameters such as work stealing, or run termination.

As prerequisite for running GeantV simulation, the user will need to create a configuration, which can be tuned using different set of parameters such as event buffer size, vector size or several GeantV-specific concurrency options. GeantV allows steering a fixed number of co-

operating working threads managed by "propagators", having as objective to increase data and processing locality while reducing contention. GeantV can use both threads or tasks, cluster threads on the same processor in NUMA-aware sub-clusters and use knowledge of the hardware topology to configure the partitioning of resources.

Our handles for tuning are configurations of jobs to be run on nodes providing a matrix of tunable parameters, based on data provided by HPC scheduler and nodes characteristics that could be evaluated through aggregated matrix of fitness functions (Figure 2).

A further improvement could be introduced in terms of event management, based on evolutionary algorithm parametric tuning of simulations configurations and event dispatching parameters (Figure 1). Using event balancing policies in a multi-node environment can be a strategy helping to reduce the expensive "tails" of miss-scheduled jobs on less efficient processing nodes.

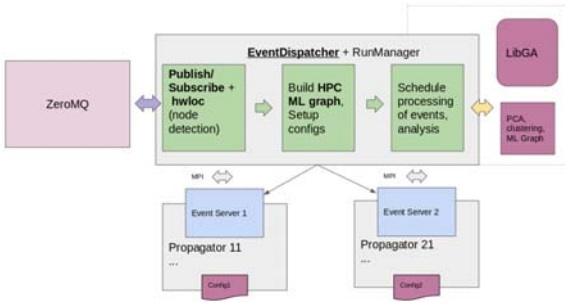


Figure 1: Event distribution schema

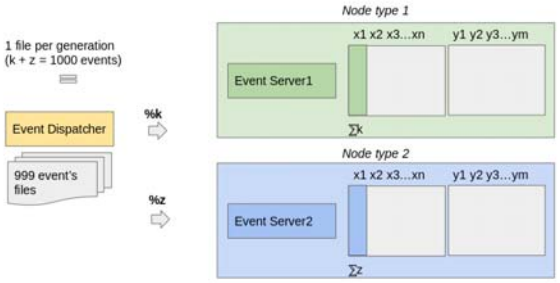


Figure 2: Data collection for tuning

3. Genetic optimization and multi-variate analysis (MVA) operators

The genetic algorithm (GA) method is one of the widely used evolutionary methods for various optimization problems. The theory of genetic algorithms was a wide subject of research for the last decades. The most basic model that allows accurate mathematical formulation of the genetic algorithm is the so-called "Simple model of Genetic Algorithm" (SGA)[7] that could be used as a prototype for an evolutionary system. This model describes the genetic algorithm as a dynamical system with accurate mathematical definitions, well studied in the literature.

In the modeling of GA as a Markov chain, the states of the evolutionary system are populations, while state transitions are handled by genetic operators: selection, crossover and mutation [8]. Mutation ensures that the Markov chain is connected, therefore there is an unique equilibrium distribution over populations. The probability to produce a particular population in one generation depends only on the previous generation and external influencing factors. This randomized process is described by a Markov chain, characterized by a transition matrix $\mathcal{T}_{\vec{q},\vec{p}}$ from the population $\vec{p}$ to the population $\vec{q}$, where in the population vector

$$\{\vec{p} = (p_1, p_2, ..., p_m)^t, \quad 0 \leq p_\alpha \leq 1, \ \sum_{\alpha=1}^{m} p_\alpha = 1\},$$

component p_α is the probability of occurrence α-th individual in the population. We have a population of m different types of individuals in the sample space Ω.

Dynamical systems describe the evolution of individuals in the space of finite dimension of possible populations of fixed size m, where m is number of measurements during the experiment. While rethinking the genetic algorithms as a discrete dynamical system, many interesting mathematical objects like fixed points could be found. These objects are apparently not only generic for simple genetic algorithms, but also general for optimization problems.

Let's briefly recall the results presented in [7] and establish the possible links with the task of optimizing our parameters. The genetic operators (selection, crossover and mutation) are acting in the space $\Lambda = (p_1, p_2, ..., p_m)^t$ of the population vectors which can consider as a $(m-1)$-dimensional simplex (a hyper-tetrahedron).

The full genetic operator $\mathcal{G}_\alpha(\vec{p})$ is a certain probability of producing individual α in the next generation if the previous population was $\vec{p}$ and define map $\mathcal{G} : \Lambda \to \Lambda$, where $\mathcal{G}(\vec{p}) = \prod_{\alpha\in\Omega} \mathcal{G}_\alpha(\vec{p})$, and $\mathcal{G}(\vec{p}) \in \Lambda$ could be considered as heuristic function. $\mathcal{G}(\vec{p})$ is GA procedure on $\vec{p} \in \Lambda$ and the map $\mathcal{G}$ is actually the composition of three different maps: selection, mutation and crossover.

Let define genetic selection operator $\hat{\mathcal{F}} : \Lambda \to \Lambda$, where $\mathcal{F}(\vec{p}) = \prod_{\alpha\in\Omega} \mathcal{F}_\alpha(\vec{p})$ and the α-th component, $\mathcal{F}_\alpha(\vec{p})$, represents the probability of the appearance of an individual of type α if the selection is applied to $\vec{p} \in \Lambda$. A selection operator chooses individuals from the current population using the cost function vector, $\vec{f} = \{f_\alpha\} \in R^m$, where $f_\alpha = f(\alpha)$, $\alpha \in \Omega$. This generic type of selection collects elements with probability proportional to their fitness. This corresponds to a heuristic function

$$\mathcal{F}(\vec{p}) = \frac{\mathrm{diag}\,(f)\cdot\vec{p}}{\vec{f}^{\,t}\cdot\vec{p}},$$

where $\mathrm{diag}\,(\vec{f})$ is the matrix with entries from $\vec{f}$ along the diagonal and zeros elsewhere.

The mutation operator $\hat{\mathcal{U}} : \Lambda \to \Lambda$ is an $m \times m$ real valued matrix with (α, β)-th entry $u_{\alpha,\beta} > 0$ for all α, β, and $u_{\alpha,\beta}$ represents the probability that individual $\beta \in \Omega$ mutates into $\alpha \in \Omega$. Then $(\hat{\mathcal{U}} \cdot \vec{p})_\alpha$ is the appearance of an individual of type α after applying a mutation to the population $\vec{p}$.

Crossover operator $\hat{\mathcal{C}}$ is defined as: $\Lambda \to \Lambda$,

$$\mathcal{C}(\vec{p}) = (\vec{p}^{\,t}\cdot\hat{C}_1\cdot\vec{p}, ..., \vec{p}^{\,t}\cdot\hat{C}_m\cdot\vec{p})$$

where $\hat{C}_1, ..., \hat{C}_m$ is a sequence of symmetric non-negative $N \times N$ real-valued matrices. Here $\hat{C}_\alpha(\vec{p})$ represents the probability that an individual α is generated by applying the crossover to population $\vec{p}$.

Combining the selection, mutation and crossover maps we obtain the complete operator $\hat{\mathcal{G}}$ for the genetic algorithm (GA map)

$$\hat{\mathcal{G}} : \Lambda \to \Lambda, \quad \hat{\mathcal{G}}(\vec{p}) = \hat{\mathcal{C}} \circ \hat{\mathcal{U}} \circ \mathcal{F}(\vec{p}). \tag{1}$$

If we know the heuristic function $\mathcal{G}$, we can write the transition matrix which is stochastic and based on the probability of transforming the population $\vec{p}$ into the population $\vec{q}$:

$$\mathcal{T}_{\vec{q},\vec{p}} = \bar{m}! \prod_{\alpha\in\Omega} \frac{(\mathcal{G}_\alpha(\vec{p}))^{\bar{m}q_\alpha}}{(\bar{m}q_\alpha)!} \tag{2}$$

where $\mathcal{G}_\alpha(\vec{p})$ is probability of producing individual α in the next generation and $\bar{M}q_\alpha$ is the number of copies of individuals α in the population $\vec{q}$, $\bar{m}$ is the size of the population.

The convergence properties of the simple genetic algorithm evolution schema was properly explored in [9]. [10] has shown that the convergence rate of the genetic algorithm is determined by the second largest eigenvalue of the transition matrix (2). The details of the proof was performed for diagonalizable transition matrices and transferred to matrices in Jordan normal form.

For the optimization of the GeantV simulation, we identify a set of optimization parameters important for the performance of particle transport simulations (e.g. the size of vector of particles to be transported or other significant design features) and build the data matrix of size $m \times n$

$$\hat{X} = \{X_{\alpha,i}\} = \{(\vec{x}_\alpha)_i\} = \{\vec{x}_\alpha\}, \tag{3}$$

which contains the values of these parameters. In this matrix index i enumerates the tuning parameters ($i = 1, ..., n$) and index α enumerates the number of measurements of the parameters ($\alpha = 1, ..., m$ for m measurements), while in terms of GA the matrix are described through m-samples of data from an n-dimensional space. In the last case m is the number of individuals in the generation and n is the size of the individual (n is the dimension of vector of genes $\vec{x} = \{x_i\}$) and the population vector is constituted by $(\vec{x}_1, \vec{x}_2, ..., \vec{x}_m)$.

In the data matrix representation we can associate the vector based on the measurements of the i-th parameter $\vec{x}'_i = \{(\vec{x}'_i)_\alpha\} = \{(\vec{x}'_i)_1, (\vec{x}'_i)_2, ..., (\vec{x}'_i)_m\}$, where the component $(\vec{x}'_i)_\alpha$ corresponds to the value of the i-th parameter in the α-th measurement with the population vector $(\vec{x}'_1, \vec{x}'_2, ..., \vec{x}'_n)$. In the probabilistic sample representation we can define the normalized probability distribution function $P_i(x)$ of the measurements of the i-th parameter which define the components $(\vec{p}_i)_\alpha$ of the population vector

$$\vec{p}_i = \{(\vec{p}_i)_1, (\vec{p}_i)_2, ..., (\vec{p}_i)_m\},$$

as the probability to measure of the i-th parameter value $(\vec{x}'_i)_\alpha$ in the α-th measurement. Then using the previous strategy we can associate the population vector $(\vec{p}_1, \vec{p}_2, ..., \vec{p}_n)$ with $(\vec{x}'_1, \vec{x}'_2, ..., \vec{x}'_n)$.

One of the challenges of a Markov chain is to determine the evolution of the components along an appropriate direction for faster convergence to equilibrium. Using Principal Component Analysis (PCA) allows to check the genetic algorithm parameter sensitivity and the possible correlation between parameters. For this we introduce an operator that will be based on PCA and inverse PCA noise reduction operation for a genetic algorithm's optimization of the set of parameters. Comparing classic and "uncentered" version of PCA [11], the second one had been shown ability of efficient noise cleanup in highly constrained data sets.

The main object of the "uncentered" PCA is the matrix of non-central second moments

$$\hat{T} = \frac{1}{m}\hat{X}^t \cdot \hat{X}. \tag{4}$$

Vectors $\vec{w}_j$ are eigenvectors of this matrix with the corresponding eigenvalues t_j

$$\hat{T} \cdot \vec{w}_j = t_j \vec{w}_j, \quad \vec{w}_i^t \cdot \vec{w}_j = \delta_{i,j}, \quad 1 \le i, j \le n. \tag{5}$$

We define matrix $W_{i,j} = \{\vec{w}_j\} = (w_i)_j$ that which diagonalizes $\hat{T}$ matrix

$$\hat{W}^t \cdot \hat{T} \cdot \hat{W} = \hat{\Delta}, \quad \Delta_{i,j} = t_i \delta_{i,j}, \quad \hat{W}^t \cdot \hat{W} = \hat{I}. \tag{6}$$

The $\vec{\theta}_j = \{(\theta_j)_\alpha\} = \{\vec{x}^t_\alpha \cdot \vec{w}_j\}\,(\alpha = 1, ..., m)$ is j-th "uncentered" principal component and define the matrix $\Theta_{\alpha,j} = \{\vec{\theta}_j\} = \{(\theta_j)_\alpha\}$, below and after the repeating indexes denote summation:

$$\Theta_{\alpha,j} = X_{\alpha,i} W_{i,j}, \quad \Theta^t_{i,\alpha}\Theta_{\alpha,j} = m\Delta_{i,j} = m t_i \delta_{i,j}, \quad 1 \le \alpha \le m, \tag{7}$$

For the variance of j-th "uncentered" principal component we obtain

$$\sigma^2_{\theta,j} = \frac{1}{m}\sum_{\alpha=1}^{m}\left[\sum_{i=1}^{n}(X_{\alpha,i} - \mu_i)W_{i,j}\right]^2 = t_j - \vec{\mu}^2\cos^2(\vec{\mu}, \vec{w}_j), \quad \mu_i = \frac{1}{m}\sum_{\alpha=1}^{m} X_{\alpha,i}.$$

For case of uncentered data matrix $\hat{X}$ we do not have a simple relationship between the eigenvalues t_j and the variance j-th "uncentered" principal component $(\sigma_{\theta,j})^2$ as for the centered case. However, this property is not essential for the usage of the PCA method for the GA and in this case it is convenient to apply the "eigenvalue control parameter" approximation. The idea is to use the PCA method for the Singular Value decomposition (SVD) representation of the uncentered data matrix.

We define the matrix $\tilde{\Theta}_{\alpha,j}$

$$\Theta_{\alpha,j} = \sqrt{m}\,\tilde{\Theta}_{\alpha,i}\Delta^{1/2}_{i,j}, \quad \Delta^{1/2}_{i,j} = t_i^{1/2}\delta_{i,j}, \quad \tilde{\Theta}^t_{i,\alpha}\tilde{\Theta}_{\alpha,j} = \delta_{i,j}. \tag{8}$$

From (7) we obtain the SVD representation for the uncentered data matrix

$$X_{\alpha,i} = \sqrt{m}\,\tilde{\Theta}_{\alpha,k}\,\Delta^{1/2}_{k,j}W^t_{j,i}. \tag{9}$$

If the matrix of non-central second moments $\hat{T}$ has $(n-q)$ smallest eigenvalues $t_j \ll 1$, $q+1 \leq j \leq n$ we can use the "eigenvalue control parameter" approximation and get the output data matrix $\tilde{X}_{\alpha,j}$ of rang q

$$\tilde{X}_{\alpha,i} = \sqrt{m}\,\tilde{\Theta}_{\alpha,k}\,\tilde{\Delta}^{1/2}_{k,j}W^t_{j,i} = \sqrt{m}\left(t_1^{1/2}\tilde{\Theta}_{\alpha,1}W^t_{1,i} + ... + t_q^{1/2}\tilde{\Theta}_{\alpha,q}W^t_{q,i}\right), \tag{10}$$

where the eigenvalue matrix $\tilde{\Delta}_{k,j}$ has rang q ($t_{q+1} = t_{q+2} = ... = t_n = 0$). Then we approximate $X_{\alpha,i}$ with rank n by the matrix $\tilde{X}_{\alpha,i}$ which has rank q. This is the analog of the Hotelling transformation.

Using the SVD representation we can estimate the mean square error η_q for this approximation:

$$\eta_q = \frac{1}{mn}\sum_{\alpha=1}^{m}\sum_{i=1}^{n}(X_{\alpha,i} - \tilde{X}_{\alpha,i})^2 = \frac{1}{n}\sum_{k=q+1}^{n} t_k.$$

The minimum error is obtained in the case if the matrix of non-central second moments $\hat{T}$ has $(n-q)$ smallest eigenvalues t_j, $q+1 \leq j \leq n$.

The second case we can get this approximation using a projector $\hat{P}^{(1,q)}$, which projects the data matrix $\hat{X}$ onto the subspace spanned by the principal axes with largest eigenvalues t_j, $1 \leq j \leq q$. Let define matrix $\tilde{W}_{i,k'} = \{\vec{w}_{k'}\} = (w_{k'})_i$ $(1 \leq k' \leq q)$ of the size $n \times q$ ($\{\vec{w}_{k'}\}$ are the first q largest eigenvectors (5)). The projector $\hat{P}^{(1,q)}$ is defined the following way

$$\hat{P}^{(1,q)}_{i,j} = \tilde{W}_{i,k'}\tilde{W}^t_{k',j}, \quad \hat{P}^{(1,q)} \cdot \hat{P}^{(1,q)} = \hat{P}^{(1,q)}.$$

Then the approximation $\tilde{X}_{\alpha,i}$ in (10) can be written using the projector $\hat{P}^{(1,q)}$

$$\tilde{X}_{\alpha,j} = X_{\alpha,i}\hat{P}^{(1,q)}_{i,j} = \Theta_{\alpha,1}W^t_{1,j} + ... + \Theta_{\alpha,q}W^t_{q,j}.$$

We considered a possibility to improve the convergence rate by adding to a standard set of GA operator's: selection, mutation, crossing in (1) a new operator $\hat{\mathcal{P}}$ performing "uncentered" PCA (UPCA) on the GA populations. UPCA-based genetic operator $\hat{\mathcal{G}}_{\mathcal{P}}(\vec{p}) = \hat{\mathcal{P}} \circ \hat{\mathcal{C}} \circ \hat{\mathcal{U}} \circ \mathcal{F}(\vec{p})$ allows to check the genetic algorithms parameter sensitivity and the possible correlation between parameters. We will analyze the result of the implementation of the operator on the uncentered data matrix on standard GA performance benchmarks. From the experimental output we see that as in the SGA case [10], the convergence rate of genetic algorithm depends on the eigenvalues following the highest one, and for this reason the proposed operator $\hat{\mathcal{P}}$ was applied on them. This approach was validated in the previous phase of research[12] and had been showed interesting results. Optimization of simulations by using genetic algorithm with UPCA operator is efficient in case non-heterogeneous cluster, but didn't take in consideration the specifications and architectural differences of computing nodes.

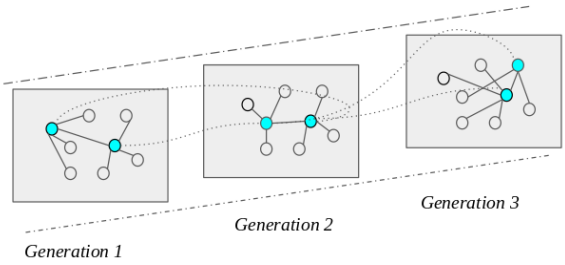


Figure 3: Multi-layer graph of HPC layer of GeantV

4. Multi-layer HPC graph in GeantV

For heterogeneous clusters, the optimization task becomes more complex, including separation of the set of nodes in sub regions with the same specification or type for independent evolutionary optimization. Taking in to account the efficiency of big data correct representation, we can use multi-graphs that provide suitable representation of the HPC system in terms of processing events. They provide opportunities to represent data more precisely and to create advance structure of data to be clustered as a next step of algorithm. Clustering algorithms try to partition a graph into several subgraphs, so that the nodes of a cluster are relatively well connected compared to the rest of the graph, while the number of edges between these clusters is limited.

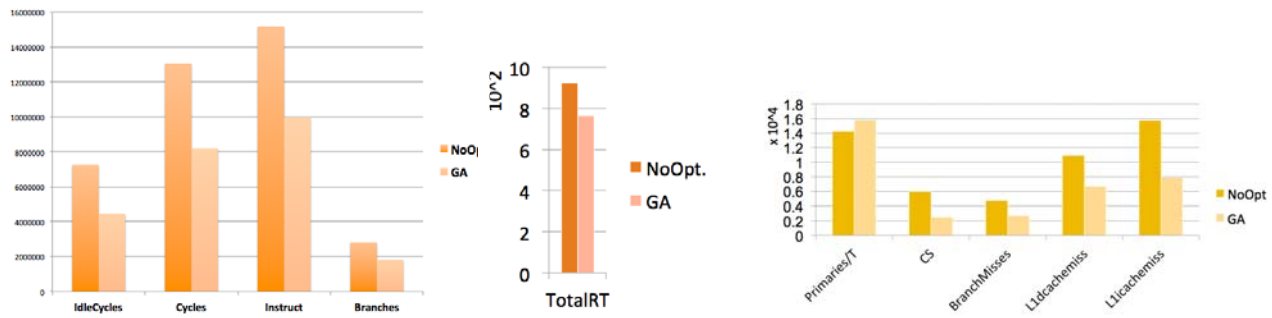
The HPC system can be seen as a "social" network where nodes are communicating with others while being exposed to events. We can define a multi-graph with edges defining the distribution of events between nodes during one genetic algorithm generation. One layer of multigraph of our system is set of batch jobs processing event files submitted to the available HPC resources (Figure 1), defined by a tensor-based computational framework based on a topology of networks with {node type (1D)} $*$ {matrix of in tunable parameters (2D)} $*$ {matrix or response functions (2D)} - 5D data set, with edges defined as an event ratio distributed by dispatchers to event servers and with graph nodes defined as a computing nodes types (Figure 3).

In case of complex heterogeneous HPC system, we are proposing to use a tensor-based computational framework to identify clusters in multiple weighted networks and using the procedure of spectral clustering, while easily separate nodes into sub-clusters using the adjacency matrix between layers.[13]

5. Results

Using methods described in section 3, we define simulation tuning parameters indexed by $i = 1, ..., n$, while index α enumerates the number of measurements of the parameters, $\alpha = 1, ..., m$ where m is a size of genetic generation. The set of parameters is defined by various performance tuning hooks available for GeantV simulation.

In Figures 4a, 4b, 4c, we stochastically tuned GeantV parameters on Core i7 machines, and we were able to attain 18% speedup of batch of job compared to the initial set. Meanwhile the results obtained for Intel(R) Xeon(R) CPU E5-2695 (Figures 5 and 6) show that by tuning these parameters we were able to achieve a reduction of the fraction of CPU usage up to 34%, providing stable memory usage and reducing the run time of the batch job up to 27%. The results achieved during this study are the proof of concept for optimizing GeantV performance using evolutionary computation tuning. The same method could be implemented for GeantV-based applications running on supercomputers and HPC clusters. The next phase of our research is to be able to tune massively parallel job sets processed in heterogeneous environment and to achieve scalability in HPC environments.



(a) (b) (c)

Figure 4: Performance tuning plots for Intel Core i7. In (a) and (c) are represented performance tuning comparison for set of parameters, while in (b) is shown batch run time ratio between not tuned and optimized simulations

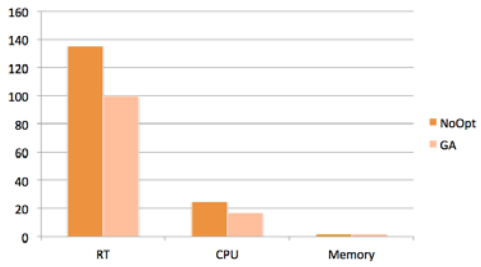


Figure 5: Perf tuning on Intel Ivy Bridge

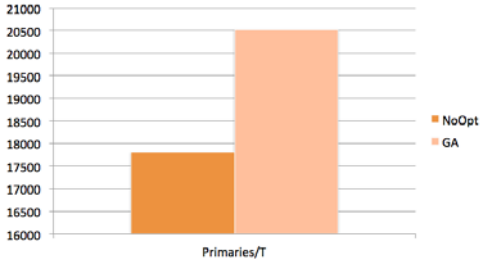


Figure 6: Primaries per time - Intel Ivy Bridge

References

[1] Apostolakis, J and al., Adaptive track scheduling to optimize concurrency and vectorization in GeantV, Journal of Physics: Conference Series, 608, 1, 012003, 2015
[2] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, Apr 2002.
[3] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," in IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577-601, Aug. 2014, doi: 10.1109/TEVC.2013.2281535
[4] S. Agostinelli and al., Geant4a simulation toolkit, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 506, Issue 3, 1 July 2003, Pages 250-303, ISSN 0168-9002
[5] Apostolakis, J and al., Towards a high performance geometry library for particle-detector simulations,Journal of Physics: Conference Series, 608, 1, 012023, 2015
[6] S. Jarp, A. Lazzaro, J. Leduc and A. Nowak, Many-core experience with HEP software at CERN openlab, J. Phys. Conf. Ser. 396, 042043 (2012), doi:10.1088/1742-6596/396/4/042043
[7] Vose, Michael D., The Simple Genetic Algorithm: Foundations and Theory, MIT Press, Cambridge, USA, 1999, 251p.
[8] J. E. Rowe. Genetic algorithm theory, Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO12, p. 917-940, New York, NY, USA, 2012. ACM.
[9] G. Rudolph. Convergence properties of evolutionary algorithms. Kovac, Hamburg, 1997.
[10] Florian Schmitt and Franz Rothlauf. On the Importance of the Second Largest Eigenvalue on the Convergence Rate of Genetic Algorithms, Proceedings of the 14th Symposium on Reliable Distributed Systems, 2001
[11] Paul Honeine, "An eigenanalysis of data centering in machine learning", preprint ArXiV ID: 1407.2904, 14p., 2014.
[12] O. Shadura, F. Carminati, Stochastic Performance Tuning of Complex Simulation Applications Using Unsupervised Machine Learning, 2016 IEEE Symposium Series on Computational Intelligence.
[13] Xiaowen Dong and Pascal Frossard and Pierre Vandergheynst and Nikolai Nefedov, Clustering with Multi-Layer Graphs: A Spectral Perspective, CoRR, abs/1106.2233, 2011, http://arxiv.org/abs/1106.2233