Understanding and mastering dynamics in computing grids: processing moldable tasks with user-level overlay

Mościcki, J.T.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
We saw that the User-level Overlay with late binding may provide important efficiency improvements for small and medium-sized workloads. The main target for grids, however, are the capacity-computing applications and very large workloads. Hence, the following question arise: what are the advantages of using the User-level Overlay with late binding for very large workloads processed over long time? In this Chapter\textsuperscript{1} we attempt answering this question by presenting a case study of a high-throughput processing system for solving a problem in theoretical physics. We also demonstrate additional benefits of the \textsc{Diane/Ganga} User-level Overlay which include a transparent access to different types of computing resources.

### 7.1 Introduction

Quantum Chromodynamics (QCD) describes strong interactions between quarks and gluons which are normally confined inside protons, neutrons and other baryons. Because the interactions are strong, the analytic perturbative expansion, where one determines exactly the first few orders of a Taylor expansion in the coupling constant, converges poorly. Thus, one commonly resorts to large-scale Monte Carlo computer simulations,

where the complete properties of QCD can be obtained numerically, up to controllable statistical and systematic errors.

In order to simulate QCD on a computer, one discretizes space and time into a 4-dimensional grid. The quark and gluon fields live respectively on the sites and bonds of this lattice\(^2\). The simulation generates a sample of the most important configurations of quark and gluon fields, evolving them one Monte Carlo step at a time. Statistical errors come from the Monte Carlo sampling, and systematic errors come from the finite lattice spacing \(a\) and finite size of the simulated 4-dimensional "box" \(N_x a \times N_y a \times N_z a \times N_\tau a\). While the three space dimensions should be in principle infinitely large, the fourth dimension defines the temperature \(T\) of the lattice: \(T = 1/(N_\tau a)\).

The majority of Lattice QCD simulations are used to study properties of the \(T = 0\) theory. Thus, all four dimensions are large, and state-of-the-art projects with, say, \(N_x = N_y = N_z \sim O(32), N_\tau \sim O(64)\), require distributing the quark and gluon fields over many CPUs which must be efficiently interconnected to maintain a reasonable efficiency. The accumulated statistics typically reach \(O(10^3 \div 4)\) Monte Carlo "trajectories".

In contrast, we are interested in high-precision measurements of some properties of QCD at finite temperature. This means that the lattice we study, of size \(16^3 \times 4\), fits into the memory of a single CPU, and that our large CPU requirements stem from the high statistics required, \(O(10^6)\) trajectories. In this case, a large pool of independent CPUs represents a cheap, efficient alternative to a high-performance cluster. This is why, in our case, using the EGEE Grid was the logical choice.

### 7.2 Problem to be solved

The physics problem we address is the following. At high temperature or density, the confinement of quarks and gluons inside baryons disappears: baryons "melt" and quarks and gluons, now deconfined, form a plasma. When the net baryon density is zero, this change is a rapid but analytic ("smooth") crossover as the temperature is raised. On the contrary, at high baryon density it is believed to proceed through a true non-analytic, first order phase transition. This change of the nature of the transition as a function of baryon density or chemical potential is analogous to the one occurring in liquid-gas transitions as a function of pressure: at low pressure water boils, and in this first-order transition, it absorbs latent heat. With increasing pressure, the transition (boiling) temperature rises and the first order transition weakens, i.e. the latent heat decreases until it vanishes altogether at a critical point, where the transition is second order. Beyond this critical pressure, the transition to the gaseous phase proceeds continuously as a crossover (with no latent heat). Correspondingly in QCD, there may exist a particular intermediate baryon density where the latent heat of the QCD phase transition vanishes and the phase transition is second-order. The corresponding temperature and baryon density or chemical potential define the so-called QCD critical point, which is the object of both experimental and theoretical searches, the former by heavy-ion collision

\(^2\)The term lattice is used in this Chapter to refer to the simulated physical system to distinguish it from the computing system used to perform the simulation.
experiments at RHIC (Brookhaven) and soon at LHC (CERN), the latter by numerical lattice simulations.

In theoretical studies, one may also consider the u, d, s quark masses as variable and investigate their influence on the order of the transition as a function of $m_u = m_d$ and $m_s$ for zero baryon density. For small enough quark masses the phase transition is of first order and corresponds to a high-temperature restoration of the chiral symmetry, which is spontaneously broken at low temperature. This chiral phase transition weakens with increasing quark masses until it vanishes along a chiral critical line, which is known to belong to the $Z(2)$ universality class of the 3d Ising model [103, 50]. For still larger quark masses, the transition is an analytic crossover. At finite density, it is generally expected that the $Z(2)$ chiral critical line shifts continuously with $\mu$ until it passes through the physical point at $\mu_E$, corresponding to the critical point of the QCD phase diagram.

What makes the study particularly interesting is that earlier results obtained for three degenerate quark flavours indicate that QCD critical point disappears [51, 52], contrary to standard expectations. This surprising result needs to be confirmed for non-degenerate quark masses, which we do here, and on finer lattices. If it turns out to be a property of the continuum QCD theory, it will have a profound impact on the QCD phase diagram. In particular, it will make it unlikely to find a QCD critical point at small baryon density.

### 7.3 Simulation model

The quark and gluon fields in the Lattice QCD simulation are mapped onto a discrete space-time $16^3 \times 4$ lattice. The simulation evolves the configurations of quark and gluon fields in a succession of Monte Carlo trajectories. The lattice is studied at 18 different temperatures around that of the phase transition. The temperatures correspond to the values of the parameter $\beta$, the lattice gauge coupling constant. What is measured is the response to a small increase in the baryon density. The signal is tiny and easily drowned by the statistical fluctuations.

A complete lattice configuration is kept in a snapshot file and the initial configurations for each $\beta$-value are called mother snapshots. Each snapshot may be evolved in Monte Carlo time by a series of iterations. The signal to noise ratio is very small and requires a large number of iterations to become significant. However, if random number sequences are different then multiple parallel Monte Carlo tracks may be used for the same $\beta$-value. The mother snapshots may be replicated and tracks use different random seeds. The tracks execute independently of one another and represent sequences of independently evolving simulation steps. The snapshot’s maturity is the number of iterations performed on that snapshot (see Fig. 7.1).

At first the replicas of the mother snapshots are identical. The subsequent iterations lead to randomization of the replicas. After a large number of iterations snapshots mature and diverge enough to contribute statistically independent simulation results. Before the randomization point is reached the snapshots are immature and only provide statistically correlated contributions. The number of iterations needed to randomize...
the lattice was not a priori known and it was estimated to be between 300 and 500 iterations (corresponding to 20-30 CPU days on a standard 2008 PC) per snapshot. This corresponds to the amount of processor time “wasted” on randomization of the lattice. The sequential overhead in this case is very large, both in absolute terms and as a fraction of entire computation. As the number of available processors varies on much shorter time scales compared to the duration of the computation at some point the number of available processors may become smaller than the number of snapshots. Then a scheduling problem arises: how to choose a subset of snapshots in order to achieve a required number of “useful” iterations before the specified deadline? Processing more snapshots than the number of available processors would result in serialization of computations. Given the long randomization time and the large number of snapshots, a naive scheduling would lead to spending all CPU time in randomizing the replicas rather than doing “useful” work. Due to the dynamic nature of grids, large fluctuations in the number of simultaneously running jobs were expected. Therefore, the system was facing the following challenges:

- adapt the scheduling algorithm so that the number of useful iterations may be maximized,
- manage the utilization of resources available in the Grid on par with the number of parallel simulation tracks,
- run autonomously over long periods of time.
7.4 Implementation and operation of the simulation system

7.4.1 Processing with HPC resources

The pre-thermalization of the Lattice QCD system was performed on a NEC-SX8 vector machine (Table 7.1) at HLRS in Stuttgart\(^3\). About 10 CPU minutes were required per Monte Carlo trajectory, and about 500 trajectories per \(\beta\)-value were produced. The fundamental reason for using vector machines in the pre-thermalization phase is the considerably higher throughput than the average node on the Grid. As finer lattice spacings are involved and the lattices get larger, exploiting fine-grained parallelism may also be beneficial. In this case a parallel architecture with low-latency interconnect is required.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Performance</td>
<td>1.2 TFlops</td>
</tr>
<tr>
<td>Processors</td>
<td>80 CPUs</td>
</tr>
<tr>
<td>Number of Nodes</td>
<td>10</td>
</tr>
<tr>
<td>Memory/node</td>
<td>128 GB</td>
</tr>
<tr>
<td>Disk</td>
<td>160 TB shared disk</td>
</tr>
<tr>
<td>Node-node interconnect</td>
<td>IXS 8 GB/s per node</td>
</tr>
</tbody>
</table>

Table 7.1: NEC-SX8 supercomputer characteristics.

The mother snapshots obtained on a NEC-SX8 vector machine were then used for the subsequent processing on the EGEE Grid which took place from April to October 2008 (and then was followed by additional runs).

7.4.2 Processing with the EGEE Grid

The QCD simulation system for the EGEE Grid was implemented with the User-level Overlay as shown in Fig. 7.2.

The master is responsible for the task scheduling and controls the order in which the snapshots are scheduled for processing to individual workers. The snapshot files are stored on the local file-system of the master and are exchanged with the worker nodes using the DIANE file transfer service. Small application plugins written in the PYTHON programming language are used to customize the DIANE framework for the needs of the Lattice QCD processing.

Each worker performs a given number of iterations and uploads the resulting snapshot file back to the master. The snapshot is then ready to be evolved further by a free worker agent. In order to avoid unnecessary network traffic, once a particular worker agent downloads a snapshot, it keeps processing it as long as possible. Therefore, the snapshot does not have to be downloaded multiple times and the worker continues the simulation using the snapshot already cached at the worker node. The worker agent

\[^3\]Nec-SX8 technical description, http://www.hlrs.de/systems/platforms/nec-sx8
Figure 7.2: Simulation system for LQCD studies. Task processing is handled by DIANE RunMaster with LQCD-specific plugins. Worker agent submission is performed manually with a help of GANJA submitter scripts, or automatically by the GANJA-based Agent Factory. Snapshot files are transferred using a built-in DIANE File Transfer Service to and from the Grid and stored in a local repository.

runs as a grid job and it has limited lifetime. The typical limiting factor is the time limit on the batch systems at the Grid sites.

Worker agents are submitted using GANJA. In the initial phases of the study the submission was done manually by the users with agent submitter scripts developed in Sec. 4.7.1. In later phases of the study the submission was performed by the Heuristic Agent Factory developed in Sec. 4.7.3.

The processing on the EGEE Grid was split in several runs. The processing workflow (Fig. 7.3) involved an active participation of the end users: the intermediate simulation results were analyzed on-the-fly by the theoretical physicists. This lead to several modifications and fine tuning of the processing including the simulation code, the number of $\beta$-values, the number of snapshots and the scheduling algorithms. The processing was also interrupted due to technical reasons such as service upgrades or hardware downtime. The processing phases are summarized in Table 7.2.

<table>
<thead>
<tr>
<th>run</th>
<th>$N_\beta$</th>
<th>$N_{\text{snapshot}}$</th>
<th>duration [weeks]</th>
<th>iterations $[\times 10^3]$</th>
<th>$N_{\text{CPU}}$</th>
<th>$T_{\text{CPU}}$ [years]</th>
<th>data transfer [TB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>400</td>
<td>11</td>
<td>300</td>
<td>4142</td>
<td>52</td>
<td>1.4</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>1450</td>
<td>9</td>
<td>700</td>
<td>21432</td>
<td>121</td>
<td>3.4</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>1063</td>
<td>3</td>
<td>267</td>
<td>12197</td>
<td>47</td>
<td>1.3</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>1063</td>
<td>8</td>
<td>266</td>
<td>12105</td>
<td>59</td>
<td>1.3</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td></td>
<td>31</td>
<td>1533</td>
<td>49876</td>
<td>279</td>
<td>7.5</td>
</tr>
</tbody>
</table>

Table 7.2: Summary of LQCD processing runs
Figure 7.3: Monte Carlo time history of a typical measured QCD observable, for example the energy, illustrating the LQCD processing steps. The observable is relaxing towards its equilibrium value and then fluctuating around it. Simulation with a supercomputer is performed to produce a mother snapshot, which then serves as a starting point for a number of grid runs with different random number initializations, yielding measurements shown by the multiple fluctuating lines.

The goal of runs 1 and 2 in the first, most critical phase of the processing, was to achieve 700,000 iterations, including the snapshot randomization, within approximately 10 weeks, in order to obtain publication-quality results. The average execution time (wall-clock elapsed on the worker node) per iteration was estimated at 1.5-2.5 CPU h on a typical 2008 PC. The size of the snapshot file (input and output of each iteration) was 10 MB. In run 1 the lattice was analyzed for a quark mass of $am = 0.0065$ with 16 $\beta$-parameters (i.e. temperatures) uniformly distributed in a value range ($5.1805, 5.1880$). In run 2 the estimation of the quark mass was refined at $am = 0.0050$, i.e. $m_s/m_{u,d} = 50$. Additional $\beta$-parameters were defined in the middle of the range and placed in between the existing values to provide more simulation points in the vicinity of the phase-transition point to the quark-gluon plasma, further referred to as sensitive region.

The reduction of the quark mass lead to longer execution time for the Monte Carlo step. This was compensated by reducing the frequency of measurements, to obtain a small overall reduction in CPU time per iteration. Run 1 and 2 were performed in parallel. The simulation parameters of run 2 were better tuned, therefore run 2 has eventually become the reference for the publication of physics results [53]. Processing runs 3 and 4 were performed in a second phase and provided more precise data for subsequent studies. The $\beta$-value range was reduced as well as the total number of snapshots.
7.4.3 Analysis of system performance

Analysis of the system performance is based on the monitoring data collected by the DIANE master. For each run a journal file is generated which contains a complete record of events that occur between the master and the workers, and which is used to extract system parameters such as the number of active workers, the number of added workers, task duration etc. All the quantities have been sampled in one hour intervals.

The evolution of run 1 is presented in Fig. 7.4, run 2 in Fig. 7.5, run 3 in Fig. 7.6 and run 4 in Fig. 7.7. Due to missing data only selected periods of each run are shown. The left vertical axis shows the size of the worker pool i.e. the number of worker agents and the number of produced iterations in each time interval. The right vertical axis provides the scale for the total number of iterations. Some exceptional events occurring during the runs, such as expiry of grid user credentials or server problems, add to natural fluctuations of the worker pool. The most important events are marked with arrows and described in the captions.

The lifetime of worker agents is limited by the batch queue limits and therefore the pool of productive workers is constantly changing. The workers which run the simulation during the time interval and successfully upload the result later are considered active. Workers which run the simulation but which do not upload the result are not considered active.
Figure 7.5: History plot of run 2 (selected period). Manual submission until 15/07. Meaning of symbols: $F_N$ – indicates the moment when the factory was enabled to keep $N$ worker agents in the pool; $E$ – workers dropped due to expired user grid credentials; $f_{\text{scale}}$ – rescaling factor for the number of iterations per hour.

active in a given time interval. This is the case of workers which were interrupted by the batch system due to the batch system time limits. In practice every worker becomes eventually inactive for a certain time before termination: a worker gets the workload from the master and runs the simulation which is interrupted by the batch system when the time limit is exceeded. This effect is called premature worker cancellation. The workers which never became active, i.e. did not manage to upload any results at all, are considered invalid.

The worker failure reasons are multiple. In a sample of 1625 invalid workers failure logs we have found $O(20)$ different failure reasons which may be broadly attributed to 3 classes: (A) incompatible processor architecture and floating-point instruction set, (B) incompatible or misconfigured system environment in the worker nodes and (C) transient connectivity problems in the DIANE processing layer. While it is possible for the class A and B to specify the resource requirements to exclude a subset of incompatible resources, it is not always efficient from the user point of view as it effectively requires long and tedious error log analysis. Another strategy for dealing with problems in class A would be to compile on-the-fly the simulation program. In our particular case this option was not possible because we used the Intel FORTRAN optimizing compiler.
Figure 7.6: History plot of run 3 (selected period). Meaning of symbols: $F_{1063}$ – factory enabled for 1063 workers in the entire period; $P$ – power failure of the master server.

which is not readily available in the Grid sites. The resource selection problem may be efficiently handled in a general way using the Heuristic Agent Factory developed in Sec. 4.7.3. Efficiency of this approach for LQCD simulations is analyzed in Sec. 7.6.

The number of performed iterations in a given time interval is proportional to the number of completed snapshots by the active workers in the pool. Each snapshot is uploaded after 3 completed iterations. The ratio between the number of active workers and the number of produced snapshots per hour is indicated by $f_{\text{scale}}$ on Fig. 7.5: $f_{\text{scale}} \simeq 1.5$ is a typical value for most of the runs while $f_{\text{scale}} \simeq 1.0$ corresponds to a larger number of faster workers being available in the grid in certain periods.

7.5 Task scheduling and prioritization

In this section we describe an application-specific scheduling which was developed for the LQCD simulation system. Its main feature is usage of knowledge of the $\beta$-parameter space to improve the total simulation throughput by ranking and prioritization of tasks to maximize the scientific content of the simulation output.

The computational complexity of the simulation (an average execution time) decreases when the temperature of the quark-gluon plasma increases (higher $\beta$ values).
This effect is related to the physical behavior of the lattice across the transition temperature. The theoretical curve of the distribution of the execution time as a function of $\beta$ should be S-shaped and monotonically decreasing, with the inflection point at the phase-transition temperature (however, an exact model function of this dependency remains unknown).

Fig. 7.8 shows the execution time of iterations for each run. The vertical bars show the execution time range between the 25% and the 75% percentile. The points show the value of the median (50% percentile). The absolute values of execution times for each run are different because the internal parameters of the simulated lattice were modified between the runs. The rather large vertical bars account for relatively broad distributions which do not allow to further constrain the transition temperature using the execution time information. The observed distributions of execution times result from the convolution of the intrinsic distribution of the amount of required computations (given by the properties of the simulated QCD lattice) with another distribution reflecting the variability of grid computing resources.

Variability of grid computing resources comes from the non-uniform distribution of
processor speeds across the grid computing nodes. Fig. 7.9 shows the distribution of nominal time to perform $10^9$ cycles for all processors used in run 4. This distribution spans almost one order of magnitude\(^4\) and is further smeared by the fluctuations which result from sharing of the same computing nodes by different jobs of many users. The amount of sharing is defined locally by the grid sites, however, it may be expected to occur frequently as almost all computing nodes in the EGEE Grid are multi-core machines: in run 4 we recorded the majority of nodes to be 8-cores (66\%) or 4-cores (25\%) and only less than 1\% to be single core.

The amount of computation required by the simulation depends on $\beta$ value and two categories may be distinguished: $\beta \leq 5.1818$, where the QCD lattice is at low temperature, characterized by many small eigenvalues of the Dirac matrix and hence a larger computing time in the associated linear solver, and $\beta \geq 5.1820$ at high temperature, where the plasma is formed and the small Dirac eigenvalues disappear. We also expect a secondary peak of the other kind in each distribution because the separation into two categories is valid only for an infinitely large lattice. This is confirmed by the task execution histogram presented in Fig. 7.10 (bottom) which shows a secondary peak at small CPU-time for low value of $\beta$.

We tried to disentangle the intrinsic and grid distributions by first considering the highest temperature, where the amount of computation required fluctuates the least. The distribution of execution times $t$ is shown in Fig. 7.10 (top). The grid variability produces a tail in the distribution, which appears consistent with an exponential form. Given that there exists an intrinsic, minimum execution time $t_0$ imposed by the hardware, a simple empirical ansatz for the distribution of execution times is then

$$t_{\text{exec}} \sim (x - 1)^\nu \exp(-cx)$$  \hspace{1cm} (7.1)

where $x = t/t_0$. This ansatz gives a reasonable description of the data at high $\beta$ (Fig. 7.10, top), with $\nu \approx 3/2$ and $c \approx 3$. It turns out that the optimal values of $\nu$ and $c$ hardly varied with $\beta$, so that we kept them fixed and considered a single fit parameter $t_0(\beta)$, reflecting the variation of the computing requirements with $\beta$ due to the physics of the problem. The fit remained rather good at all temperatures $\beta$, with $t_0(\beta)$ increasing monotonically as $\beta$ is decreased, and most steeply at the critical temperature, as expected on physical grounds.

### 7.5.1 Maturity-based scheduling

In run 1 the snapshots were dynamically prioritized based on their maturity: the snapshots with the least number of Monte Carlo iterations were scheduled before the more mature ones. Let $S_{\beta k}^s$ denote a snapshot after $k$ iterations at a given temperature $\beta$ and for initial random seed $s$. Let $S_{\beta_1}^{s_1}(k_1) < S_{\beta_2}^{s_2}(k_2)$ denote that $S_{\beta_1}^{s_1}(k_1)$ should be scheduled before $S_{\beta_2}^{s_2}(k_2)$. The maturity-based scheduling policy may be defined as:

$$S_{\beta_1}^{s_1}(k_1) < S_{\beta_2}^{s_2}(k_2) \iff k_1 < k_2.$$  \hspace{1cm} (7.2)

\(^4\)While largely approximate, the clock speed provides an acceptable estimate of processing power for CPU-bound simulations.
Figure 7.8: Median of elapsed wall-clock time of 3 iterations as a function of $\beta$-value. The vertical bars show the execution time range between the 25% and the 75% percentile. The points show the value of the median (50% percentile).

Figure 7.9: Characteristics of CPUs in the EGEE Grid. Distribution of time to complete $10^9$ CPU cycles per processor. The highest peak is in 2-3 GHz CPU clock rate range.
Figure 7.10: Distribution of task execution times (elapsed wall-clock time on the worker node) in run 4 for high $\beta$ (top) and low $\beta$ (bottom). For high $\beta$ the empirical function describes the data very well. For low $\beta$, the secondary peak at small CPU time may be explained on physical grounds, as a remnant of the high-temperature phase.
In first approximation, the objective is to evolve all snapshots keeping the spread in the iteration number as small as possible.

### 7.5.2 Scheduling in the sensitive region

After the initial analysis of simulation results, it was decided to change the range of $\beta$-parameters and the scheduling policy for run 2 to improve the convergence speed. A finer-grained sensitive $\beta$-region $R = [5.1815, 5.18525]$ around the expected plasma transition temperature was defined. Within the sensitive region $R$ the scheduling policy was to select snapshots with a smaller $\beta$-value first

$$S_{\beta_1}(k_1) < S_{\beta_2}(k_2) \iff \beta_1 < \beta_2.$$  \hfill (7.3)

Outside of the sensitive region the maturity-based prioritization was kept. Snapshots from the sensitive region were always selected before any snapshots from the outside of the region. Thus the scheduling policy in the entire range was defined as

$$S_{\beta_1}(k_1) < S_{\beta_2}(k_2) \iff \begin{cases} \beta_1 \in R, \beta_2 \notin R \text{ or } \\
\beta_1 < \beta_2 \text{ and } \beta_1, \beta_2 \in R \text{ or } \\
k_1 < k_2 \text{ and } \beta_1, \beta_2 \notin R. \end{cases}$$  \hfill (7.4)

This gives absolute priority to the sensitive region, and within that region to smaller $\beta$ values.

In runs 3 and 4 the sensitive region was expanded to include $\beta$-values below the expected phase transition point, $R = [5.1805, 5.18525]$. At the same time the $\beta$-values above the sensitive region were removed from the simulation.

### 7.5.3 Analysis of the scheduling results

Fig. 7.11 shows the maturity of the snapshots at the end of each of the runs, grouped by the values of $\beta$. The final distribution of maturity depends on the computational requirements for each $\beta$-value, the scheduling policy and runtime factors.

Results obtained for run 1 and run 2 show that the maturity-based scheduling is implemented efficiently. However, the final maturity distribution in the sensitive region is influenced by number of available processors in the system.

Scheduling in the sensitive region is based on the ordering of the snapshots (with respect to the $\beta$ parameter). Therefore, in first approximation the number of completed iterations per snapshot should be larger for smaller $\beta$-values. This applies to a system working at low regime, i.e. when the number of available processors is smaller than the number of snapshots. The effect is observed for run 4 in Fig. 7.7: more than half of the time the system is working at 500 workers or less, what corresponds to 50% of the required processing capacity.

At full capacity, the number of available processors is equal to or higher than the number of snapshots. In this case the actual ordering priority does not matter because each snapshot is processed at any time. Considering that the worker pool is constantly changing (workers join and leave quite often) the ordering of snapshots does not influence
Figure 7.11: Total number of completed iterations in the different $\beta$ ranges for all runs. The sensitive region is indicated in darker colour (red). The sawtooth pattern visible in run 2 is due to interleaved $\beta$-values added after the run started, thus completing less iterations.
the effective allocation of snapshots to workers. The final maturity of snapshots at the end of a run depends on the amount of processing in the function of $\beta$ and the distribution of processing power of the workers.

### 7.6 Analysis of adaptive resource selection

In run 1 (Fig. 7.4) and in the first part of run 2 until 15/07 (Fig. 7.5), the workers were added to the pool with manual job submission by users without adhering to any particular submission schedule. In the remainder of run 2 and in runs 3,4 (Figs. 7.6, 7.7) the submission was controlled by the Heuristic Agent Factory (HAF) developed in Chapter 4. HAF was enabled to maintain $N$ active workers in a pool as indicated by $F_N$ events. When HAF is enabled, the number of invalid workers is less scattered and under better control. The resource selection algorithm implemented by HAF reduces the number of invalid worker agents and thus reduces the number of failing jobs flowing in the Grid which have negative impact on scheduling due to premature worker cancellation as described in Section 7.4.3. However, a small background of invalid workers exists and it is a feature of the selection algorithm were a fraction of jobs are submitted to random CEs via the generic slot.

In the case of exceptional events, such as expiry of grid credentials ($E$), the number
of invalid workers rapidly increases as the number of active workers falls sharply. The number of compatible resources suddenly drops to zero as all new submissions fail and all running workers are interrupted. Such events have a similar impact on the system, independently of whether the workers are submitted manually or via the HAF.

The distribution of the worker pool size in run 2 is shown in Fig. 7.12. In the manual submission mode, the distribution shows a large scatter below the optimal threshold for $N_{\text{workers}} = N_{\text{snapshots}} = 1450$, which indicates under-provision of the worker agents to the system. In case of HAF, the three clear peaks of the distribution correspond to three stages of the run, as indicated by the events $F_{800}, F_{1200}$ and $F_{1450}$. When HAF is enabled and resources are available, then the number of active workers quickly converges to the requested level and is maintained for an extended period of time.

The HAF may not maintain the required level of the workers in the pool if there are not enough resources in the Grid. In run 4, the processing enters a low regime phase $R$, where the amount of available resources is clearly below the optimal target of $N_{\text{workers}} = N_{\text{snapshots}} = 1063$. Under such conditions the resource selection based on best fitness allows to reduce the number of invalid workers as compared to the manual submission.

Occasionally the HAF leads to oversubmission of worker agents, e.g. $F_{1063}$ in run 4. The default policy of the Agent Factory is to fill up available computing slots within a Computing Element until the worker agents start queuing. If many computing slots become available at the same time in a larger number of Computing Elements, then a
Figure 7.14: Application model deployed on the EGEE Grid (a) compared with low-level parallelism possible with TeraGrid resources using shared-memory OpenMP threads (b) and distributed-memory MPI processes (c). Each simulation track consists of a sequence of tasks which are managed by the User-level Overlay. The realization of a single task differs in all three cases.

large number of queuing jobs suddenly start running and the worker pool could grow beyond the requested size.

The HAF submits workers more efficiently such that a larger number of active workers is added to the system in a time unit. In run 2, the HAF yields an average stream of 45 active worker agents per hour as compared to 23 active worker agents per hour in manual submission mode as visible in Fig. 7.13. The distribution of the number of added workers in a unit of time displays a clear difference in the submission patterns.

Finally, the HAF allows the system to work autonomously and a drastic cut in the time needed for human operation was observed during the study. Only seldom incidents, such as power cuts, needed manual interventions.

7.7 Exploiting low-level parallelism for finer lattices

Our study indicated that for physical quark masses the QCD transition is a crossover at $\mu = 0$ which becomes even softer as a small chemical potential is switched on. It is now most interesting and important to repeat these calculations on finer lattices, in order to see whether this behavior of the chiral critical surface is also realized in continuum QCD.

Increasing the lattice size for our application would require an efficient strategy of
handling $O(100)$ distributed simulation tracks each being a locally-parallel job with $O(30)$ processes. First, experiments with Ganga/Diane overlay extended to use multicore processors showed feasibility of using OpenMP as a parallelization technique if lattice size is increased moderately. Multicore resources available in TeraGrid are accessed via a SAGA thanks to a Ganga plugin. A task corresponds to execution of several OpenMP threads in parallel (with shared memory) on the same computing node (see Fig. 7.14). The task management is handled by the User-level Overlay as in the case of the EGEE Grid. This approach allows to easily reuse existing application code without much refactoring. However, the efficiency of parallelization is limited to a relatively small number of processors – initial tests indicate that no major gain is achieved beyond 4 parallel OpenMP threads per task.

A similar approach, which may be investigated in the future, consists of enabling the application for parallel processing with MPI. The management of tasks and jobs may be done in a similar way as in the case of OpenMP using the User-level Overlay. SAGA-based Ganga plugin could provide simultaneous allocation of groups of worker agent jobs and Diane framework for subsequent task scheduling such that one group of worker agents handles one task at a time. One task would correspond to a set of MPI processes (with distributed memory) running on a group of closely-connected computing nodes (see Fig. 7.14).

7.8 Summary

We demonstrated that a User-level Overlay may be used efficiently, yielding $O(10^3)$ speedup, to produce complex scientific results and to process very large workloads over long periods of time (capacity computing). The Lattice QCD application described in this Chapter has the following features: a large granularity (one iteration took over an hour), a small I/O requirement (10 MB per hour or less), and a robust single-CPU code. These features are not typical for other Lattice QCD applications which often simulate too many degrees of freedom to be handled by a single CPU.

For our application, the management and scheduling of $O(10^3)$ independent simulation tracks was advantageously handled by Ganga/Diane User-level Overlay. With limited high-level scripting, we plugged into the Master service scheduling algorithms which exploited the knowledge of the status of the simulated lattice. Dynamic resource selection based on the application feedback was automatically provided by Heuristic Agent Factory and allowed to reduce wasted resources to $O(10\%)$. With the exception of external events such as service power outage or minor manual interventions such as upgrade of the application code or renewal of user credentials to the grid, the system operated autonomously for several months and showed exceptional stability.

For this application the EGEE Grid enabled to obtain scientific results faster and at a lower cost that using massively-parallel computing facilities such as High-Performance Computing centers. While comparable CPU resources are commonly available there, they are usually packaged within expensive massively parallel machines. High-availability computing nodes and high-bandwidth network interconnects are a more expensive option than less reliable, commodity elements used in grids. Grids also enable resource
sharing to lower the total cost of ownership for resource providers, as the computing resources may be leased or borrowed more flexibly, according to current needs. While for this application the initial pre-thermalization phase was conducted using a supercomputer in a HPC center, for the bulk of the simulations a very large pool of loosely connected, heterogeneous PCs provided an adequate, cheaper platform.

Finally, it is worth mentioning that the study described in this Chapter has shown that there is no QCD chiral critical point at temperature $T$ and small quark chemical potential $\mu_q$ satisfying $\mu_q/T < \mathcal{O}(1)$, on a coarse lattice with 4 time-slices, with a strange quark having its physical mass.