Parallel Processing Letters
© World Scientific Publishing Company

QUASI MONTE CARLO INTEGRATION IN GRID ENVIRONMENTS: FURTHER LEAPING EFFECTS

HEINZ HOFBAUER, ANDREAS UHL, PETER ZINTERHOF*
Department of Scientific Computing, Salzburg University
Jakob-Haringerstr.2, A-5020 Salzburg, AUSTRIA.

Received November 2005
Revised (revised date)
Communicated by Marian Vajtersic

ABSTRACT

The splitting of Quasi-Monte Carlo (QMC) point sequences into interleaved substreams has been suggested to raise the speed of distributed numerical integration and to lower the traffic on the network. The usefulness of this approach in GRID environments is discussed. After specifying requirements for using QMC techniques in GRID environments in general we review and evaluate the proposals made in literature so far. In numerical integration experiments we investigate the quality of single leaped QMC point sequence substreams, comparing the respective properties of Sobol', Halton, Faure, Niederreiter-Xing, and Zinterhof sequences in detail. Numerical integration results obtained on a distributed system show that leaping sensitivity varies tremendously among the different sequences and we provide examples of deteriorated results caused by leaping effects, especially in heterogeneous settings which would be expected in GRID environments.

Keywords: Quasi-Monte Carlo Integration, Grid Computing, Leap-Frog Technique, Subsequences.

1. Introduction

High dimensional numerical integration problems may require a significant amount of computation power. Therefore, substantial effort has been invested in finding techniques for performing these computations on all kinds of parallel architectures (see [1,2,3,4] for an exhaustive overview). In order to minimize the communication amount within a parallel system, each processing element (PE) requires its own source of integration nodes. Therefore, the aim is to investigate techniques for using separately initialized and disjoint sets of integration nodes on a single PE.

Currently, the most efficient numerical techniques for evaluating high-dimensional integrals are based on Monte Carlo and quasi-Monte Carlo techniques [5]. Whereas in the Monte Carlo (MC) case the integration nodes are produced by a random number generator (RNG), low-discrepancy point sets and sequences (e.g. (t,m,s)-nets

*Corresponding author: Andreas Uhl, e-mail: andreas.uhl@sbg.ac.at or (t,s)-sequences [6]) are employed in quasi-Monte Carlo (QMC) algorithms. QMC techniques improve the probabilistic error bounds of MC techniques especially in higher dimensions. Nevertheless, these techniques are related [7] since a full period random number sequence may be seen as a low-discrepancy point set (e.g. a rank-1 lattice rule in the case of a linear congruential generator) as well.

GRID environments exhibit challenging properties for numerical integration techniques. This type of computing facility potentially shows extreme heterogeneity in terms of PE speed and network connections, moreover the available computing resources may change in time even during ongoing computations (e.g. new machines may become available or others may get lost due to network problems or maintenance shutdowns, other users may start computations on the same hardware, etc.). Li et al. [8] discuss the use of a GRID service called Integration Service for solving multivariate integration problems, where the PARINT package [1] is used as integration engine.

Therefore, it is not only difficult but impossible to predict the amount of integration nodes required on a single PE under such conditions. Additionally, in practice it is usually not possible to determine a priori the number of integration nodes N necessary to meet a given error requirement. As a consequence, it is of great importance that N may be increased without losing previously calculated function values. It is clear that techniques for providing integration nodes on the single PEs need to be very flexible under these circumstances.

One possibility is to generate separately initialized and disjoint substreams of a given (sequential) sequence of integration nodes. Among other suggestions made in literature in this context, we have investigated leaped substream parallelization techniques for (t,s)-sequences in previous work [9,10,11] and have discovered several shortcomings and problems when applied in parallel and distributed QMC integration.

In this work, we extend the focus of our work on leaped substream parallelization to cover the following issues:

- So far, only the leaping of Sobol' and Niederreiter (t,s)-sequences has been investigated. Here we additionally treat Faure, Halton, Weyl, and Niederreiter-Xing sequences.
- So far, only the quality differences among differently leaped streams have been investigated systematically. Here we additionally treat the quality differences among equally leaped but differently initialized substreams and the respective implications for the use in distributed computations.
- So far, only a moderate amount of hardware heterogeneity has been investigated. Here we reflect potential GRID properties and additionally treat a case where computing capacity varies by a factor of 10^3 and compare the result to more classical environments.

In Section 2 we discuss strategies for using QMC techniques in GRID environments. Section 3 shortly reviews the QMC computation of improper integrals (which is being used as application case). Section 4 is the main part of this work where we report and discuss experimental integration results using various QMC node sets. Section 5 concludes this work and provides outlook to future work in this direction.

2. QMC Techniques in GRID Environments

GRID environments exhibit a potentially high heterogeneity in terms of network capacity (i.e. bandwidth and latency) and computing speed (memory capacity, cache sizes, processor speed). In addition to that, these environments are error prone with respect to broken network links or failing PEs. Moreover, additional resources may become available during an ongoing simulation which should be used to optimize resource consumption. As a consequence, the following requirements should be met by a QMC technique employed in a GRID environment:

- Variety in computing speed requires dynamic load balancing capability.
- Variety in network capacity requires load balancing strategies without central organization and a minimal number of control messages exchanged among the computing nodes.
- Failure in hardware resources requires tolerance to lost partial results.
- Additional resources becoming available require a possibility to assign workload to these resources (i.e. by redistributing or redefining workload).

In addition to that, error bounds and computation results should preferably carry over from sequential execution. If the QMC point sets differ between sequential and parallel execution, the quality of the results needs to be investigated thoroughly. Reproducibility is as well an important issue to be considered.

So far, two entirely different strategies have been discussed in literature to employ QMC sequences in parallel and distributed environments.

1. Splitting a given QMC sequence into separately initialized and disjoint parts which are then used independently on the PEs. This strategy comes in two flavors:
   - **Blocking**: p disjoint contiguous blocks of maximal length l of the original sequence are used on the PEs. This is achieved by simply using a different starting point on each PE (e.g., PE$_i$, $i = 0, \ldots, p-1$, generates the vectors $\mathbf{x}_{il}, \mathbf{x}_{il+1}, \mathbf{x}_{il+2}, \ldots, \mathbf{x}_{il+l-1}$). In case a large number of smaller blocks is used index j is assigned dynamically to PE$_i$ which generates the vectors $\mathbf{x}_j, \mathbf{x}_{j+1}, \ldots, \mathbf{x}_{j+l-1}$ (where j is incremented in steps of size l to avoid overlap).
   - **Leaping**: interleaved streams of the original sequence are used on the PEs. Each PE skips those points consumed by other PEs (*leap-frogging*)

(e. g. employing p PEs, PE$_i$, $i = 0, \ldots, p-1$, generates the vectors $\mathbf{x}_i, \mathbf{x}_{i+p}, \mathbf{x}_{i+2p}, \ldots$).

2. Using inherently independent sequences on the different PEs (denoted as "parametrization" which can be realized for example by randomizations of a given QMC sequences).

Blocking has been suggested in many application focused papers. Mascagni and Karaivanova [12] propose to use disjoint contiguous blocks from Halton, Faure, and Sobol' sequences in the context of solving sparse systems of linear algebraic equations. Numerical experiments are carried out on a homogeneous cluster using static load distribution. In a second paper [13] the same authors use the suggested techniques for computing extremal eigenvalues, again a QMC sequence is "neatly broken into same-sized subsequences" by blocking. The authors point out that this simple strategy can not be employed in general for all types of simulation settings. Alexandrov et al. [14] use scrambled Sobol' and Halton sequences to solve certain linear algebra systems. They discuss static and dynamic load balancing and point out the importance of efficient dynamic load balancing in GRID environments. Load balancing is done by dynamically distributing chunks (i.e. blocks) of relatively small size to avoid unevenly sized chunks. Techniques for efficiently generating non-adjacent chunks an a single PE are discussed in this paper. Tests are carried out on homogeneous and heterogeneous systems; in the latter case MPICH over Globus-2 GRID software is used. Li and Mascagni [15] propose to extend techniques used in GRID-based Monte Carlo methods, e.g. the *N-out-of-M* scheduling strategy, to QMC sequences by using scrambled quasi random sequences. Furthermore, known statistical properties of MC carry over to scrambled quasi random sequence and thus allowing partial result validation and intermediate value checking. Wan et al. [16] present a parallel strategy for pricing multidimensional American options. In the first stage, the QMC sequence is generated by independently computing equally sized blocks on the PEs using static load distribution. For the second stage two strategies, one being the stochastic mesh method which involves a backward recursion, for data distribution are compared which both correspond to distributing the original sequence in blocks of different size in different manner across the PEs. Tests are conducted on a SGI Onyx machine. Schürer [17] employs equally sized blocks of (t,m,s)-nets on the PEs when comparing QMC integration techniques to adaptive cubature rules. A SGI Power Challenge is used as a test platform. In previous work [9] we have conducted experiments with blocking Niederreiter (t,s)-sequences where large disjoint blocks are used on the PEs. Good reliability of the results has been observed in homogeneous and (simulations of) heterogeneous environments (tests conducted on a SGI Power Challenge). We have also provided theoretical evidence for this good behavior by showing that discrepancy estimates of arbitrary blocks do not degrade as compared to estimates of entire (t,s)-sequences [10].

Leaping has been discussed much more controversial in literature than blocking. Bromley [18] describes a leapfrog parallelization technique to break up the Sobol' sequence into interleaved substreams in an efficient manner. We have generalized this idea to all types of binary digital (t,s)-sequences [10] in earlier work. Based on these techniques, Li and Mullen [19] use a leapfrog scheme for (t,m,s)-nets to solve financial derivative problems. However, severe problems occur with leapfrog parallelization especially in case of processor speed heterogeneity which results in QMC point sets which do not correspond to sequential computation. Initial results showed that single (t,s)-sequence substreams with leaps of the form 2^n lead to extremely poor numerical integration results whereas this is not the case for leaps of the form 2^{n+1} [9]. Using leaped substreams parallelization in a heterogeneous processor speed environment therefore may lead to severely degraded results as compared to sequential execution when this form of leaping is employed. Different PEs consume a different number of integration nodes and so the poor results of using single substreams are propagated to the parallel results if no synchronization among PEs is performed [11,9,10]. We have also provided theoretical evidence for the observed effects by showing the discrepancy estimated of leaped substreams to be significantly larger as compared to the original sequences [10]. It has also turned out that not only 2^n type substreams are affected by poor quality but these effects occur for many forms of leaps and are highly unpredictable [11,10].

Parametrization has been proposed as a QMC parallelization strategy by two groups independently. DeDoncker et al. [20,21,22] propose randomized (Korobov) lattice and Richtmyer rules (which are a special type of Weyl sequences), and discuss load distribution strategies for homogeneous and heterogeneous architectures [23]. Results are provided for both, homogeneous and heterogeneous environments, and in both cases result accuracy and execution efficiency was reported to be very well. Ökten and Srinivasan [24] propose to use Halton and scrambled Halton sequences with leaped base sequences on different PEs. Excellent theoretical error estimations are provided and also experimental results for homogeneous as well as for heterogeneous environments exhibit high quality. Parametrization is also compared to blocking and leaping in this work and advantages and disadvantages of the three schemes are analyzed for different application scenarios. Srinivasan [25] confirms the findings of the latter paper and refines the comparison of the three parallelization strategies based on simulation results for pricing financial derivatives.

Based on the requirements for a QMC technique to be useful in GRID environments stated before we try to assess the effectiveness of the three parallel QMC techniques proposed in literature.

- **Blocking**: Two flavors of blocking are discussed. In the first variant, the QMC sequence is partitioned into small blocks which are dynamically distributed among the PEs. Whereas this technique uses QMC node sets almost identical to sequential execution and can handle all types of changing resource scenarios and heterogeneity quite well, it requires the frequent exchange of control messages and is therefore not suited for GRID environments. The validity of this assessment of course depends on the relation between block size and the communication possibilities in the actual GRID environment. In the second blocking variant, one large block is assigned to each PE at the start of the computation. Since the number of QMC points required on each PE is not known a priori the block size needs to be chosen large enough to avoid a PE to exceed the number of available points in its block (exceeding the number would then result in overlap of the blocks which of course degrades the final result). On the other hand, if the blocks have been selected much too large, a significant number of points may not be consumed on slow PEs and the overall point set used exhibits large "gaps" as compared to the sequential case which potentially threatens result accuracy (although the results available so far concerning this effect do not seem to be very severe). Choosing the block size appropriately is therefore a critical issue in this approach. The same considerations of course apply as well if a PE fails. In the case where a specific QMC point set with a limited number of points has been distributed among the available PEs at the start of the computation (e.g. a (t,m,s)-net), handling additional resources is fairly complicated. On the other hand, in the case of using infinite sequences only the next large block in the sequence not being assigned to a PE so far has to be assigned to a PE which has become available during the computation. Although some of the potential problematic effects (like significant block overlap or large gaps) have not been investigated systematically, the currently available results indicate reliable behavior. The use of large blocks is therefore an interesting option for GRID environments.

- **Leaping**: Contrasting to the blocking case, there is no need to specify a number of points required on each PE since each substream may deliver an infinite number of points in principle. Therefore, there is no danger of running short of points. Also, substream overlap can not occur. In the case of homogeneous environments where care is taken that each PE consumes an equal share of QMC points, a result identical to sequential execution is easily obtained. Note that this is not the case for blocking due to the problems with choosing a good block size (except in case the number of points required is known in advance – which is rarely the case). The situation changes drastically in heterogeneous environments: in case of different PEs consuming a different number of QMC points it has been shown that depending on the type of load imbalance more or less severe degradations in result accuracy are observed. We will discuss further results in this context in Section 4. The same considerations (with even more pronounced effects) of course apply as well if a PE fails. When additional resources become available the classical leaping scenario can not handle this situation properly – usually a QMC node set is partitioned into J interleaved substreams if J PEs are available. There is no additional substream available in this scenario. A way to handle this situation is to partition a given QMC point set into $I > J$ substreams in case of J PEs are available. The $I - J$ substreams are not used by default but kept as additional work share in case additional PEs become available. However, neither empirical nor theoretical results are available so far to assess the quality of corresponding results. The use of leaping in GRID environments may be therefore accompanied with problematic side effects which endanger a flexible and transparent use. This will be discussed in more detail in Section 4. In addition to that, the most important advantage of leaped substream parallelization as compared to blocking (i.e. in case of synchronized execution the used point set corresponds to the sequential case) does not apply in GRID environments due to the heterogeneity.

- **Parametrization**: The most important difference (and also disadvantage) of parametrization as compared to blocking and leaping is that the QMC point set used in parallel or distributed computation does not correspond to a single (sequentially used) point set. Therefore, the investigation of the results' quality when using this technique is of great importance since it is not clear a priori how results from different point sets will interact in the final result. The findings so far indicate a good quality of the results based on theoretical estimates and empirical tests. Due to the use of de-facto independent QMC point sets on the PEs load balancing at low cost comes for free (each PE generates as much points as it requires locally) and the same is true for reacting to changes with respect to available resources. Therefore, parametrization is a well suited approach for GRID environments provided the quality of the results can be guaranteed. A disadvantage is that knowledge about the reliability of the results is restricted so far to Halton sequences and (Korobov) lattice rules. An advantage of using randomized QMC techniques in general is that error control techniques like variance reduction are integral parts of this approach. This also holds true for their use in GRID environments.

3. QMC Computation of Improper Integrals

The basic concept of any method for numerical integration is to approximate the integral by a finite sum, such that

$$I(f) := \int_{I^s} f(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n) =: I'_N(f) \tag{1}$$

where x_n are suitably chosen integration nodes and I^s is the unit interval. To identify suitable, i.e. uniformly distributed, point sets $\{x_n\}$ the star discrepancy is defined as

$$D_N^* := D_N^*(x_1, \ldots, x_n) = \sup_{J \in \mathcal{F}} \left\| \frac{\#\{x | x \in J\}}{N} - m(J) \right\| \tag{2}$$

where $\mathcal{F}$ is the family of all subintervals of the form $J = \prod_{i=1}^{s}[0, t_i) \in I^s$ with volume $m(J)$. The approximation error

$$E_N(f) := |I'_N(f) - I(f)| \tag{3}$$

depends on D_N^* and the variation $V(f)$ of the function f in the following way (Koksma-Hlawka inequality [6]):

$$E_N(f) \leq V(f) D_N^*. \tag{4}$$

Consequently, point sets exhibiting low discrepancy values are attractive candidates to be used in numerical integration. The QMC approach chooses the point set $\{x_n\}$ from a low-discrepancy sequence.

QMC techniques have been considered to evaluate improper integrals in recent years (e.g. [26]) – for numerical integration of such types of integrands we use a generic method introduced recently by Zinterhof [27] (which can employ any type of integration nodes in principle but delivers decent error bounds for low-discrepancy integration node sets). For a given function $f(x)$ and a $B > 0$, Zinterhof specifies the functions $f_B(x)$ and $\hat{f}_B(x)$ as

$$f_B(x) = \begin{cases} f(x) & |f(x)| \leq B \\ 0 & |f(x)| > B \end{cases} \tag{5}$$

$$\hat{f}_B(x) = \begin{cases} 0 & |f(x)| \leq B \\ f(x) & |f(x)| > B. \end{cases} \tag{6}$$

The Class $C(\beta, \gamma)$ of s-variate functions $f(x_1, \ldots, x_s)$, $0 \leq x_i \leq 1$, $i = 1, \ldots, s$, consists of all functions which fulfill

$$\text{a)} \quad I(|\hat{f}_B|) \quad = O(B^{-\beta}) \text{ for some } \beta > 0 \tag{7}$$

$$\text{b)} \quad V(f_B) \quad = O(B^{\gamma}) \text{ for some } \gamma \geq 1. \tag{8}$$

It is shown that if $f(x) \in C(\beta, \gamma)$, $\mathfrak{x} = (x_1, \ldots, x_s)$, and if the discrepancy of the set of nodes is D_N^*, then for $B = {D_N^*}^{-1/(\beta+\gamma)}$ the estimate

$$I(f) = \frac{1}{N} \sum_{n=1}^{N} f_B(\mathfrak{x}_n) + O({D_N^*}^{\beta/(\beta+\gamma)}) \tag{9}$$

holds, where $I(f) = I(f_B) + I(\hat{f}_B)$. Finally Zinterhof shows that when the truncation parameter B takes on the form

$$B = {D_N^*}^{-1/(\beta+\gamma)} \tag{10}$$

the integration error attains its minimum of

$$E_N \leq K(f) {D_N^*}^{\beta/(\beta+\gamma)} \tag{11}$$

where $K(f)$ is a constant depending on f. Consequently, the numerical strategy is to compute $\frac{1}{N}\sum_{n=1}^{N} f_B(\mathfrak{x}_n)$ upon selection of a suitable B.

4. Experiments: Leaped Substreams

4.1. QMC Node Sets

For generating the Sobol', Halton, Faure and Niederreiter-Xing sequences we use the implementation of the "High-dimensional Integration Library" HIntLib.[†] The descriptions of the Sobol', Faure, and Niederreiter-Xing sequences are taken from [28].

4.1.1. Sobol' Sequence

The Sobol' sequences [29] are digital (t_s, s)-sequences over F_2, where

$$t_s = \sum_{i=1}^{s} (\deg p_i - 1), \tag{12}$$

with $p_1 = x \in F_2[x]$ and p_{i+1} denoting the ith primitive polynomial over F_2 ordered by degree.

Sobol' sequences were the first known constructions yielding (t, s)-sequences for arbitrary dimensions s. They were introduced long before the theory of (t, s)-sequences over arbitrary finite fields F_b was established in [30]. However, they only exist for $b = 2$, and even in this case, the resulting t parameter is not optimal for $s > 3$. For $s > 7$, even the Niederreiter sequence, which is equally easy to implement, yields lower t-values.

For $s = 1$ the Sobol' sequence (defined by the polynomial $p_1 = x$) is a $(0, 1)$-sequence identical to the van der Corput sequence in base 2.

We use the implementation of construction 6 in [31].

4.1.2. Faure Sequence

The Faure sequences are digital $(0, s)$-sequences over F_b with b denoting a prime number (original case) or a prime power (general case) greater or equal to s. The case for b prime was shown by Faure [32], the general result is due to Niederreiter [30, Theorem 6.2].

The s infinite generator matrices $C^{(1)}, \ldots, C^{(s)}$ over F_b are defined by $C^{(i)} = (c_{jr}^{(i)})_{j,r>0}$ with

[†]Available at: http://www.cosy.sbg.ac.at/˜rschuer/hintlib/

$$c_{jr}^{(i)} = \binom{r}{j} \alpha_i^{r-j}, \tag{13}$$

where $\alpha_1, \ldots, \alpha_s$ denote s distinct elements from F_b and the conventions $\alpha^0 = 1$ for all $\alpha \in F_b$ and $\binom{r}{j} = 0$ for $j > r$ are used.

For $\alpha = 1$, the resulting matrix is the infinite Pascal matrix modulo the characteristic of F_b; for $\alpha = 0$, it is the infinite identity matrix. If $s = 1$ and $\alpha_1 = 0$, the resulting $(0, 1)$-sequence is identical to the van der Corput sequence in the same base.

Sequences with the same parameters can also be obtained using Niederreiter sequences or Niederreiter-Xing sequences with rational function fields.

We use the implementation of construction 8 in [31].

4.1.3. Halton Sequence

The construction of the Halton sequence was introduced in [33]. For a dimension $s > 0$, let $b_1, \ldots, b_s$ be integers ≥ 2. Then the Halton sequence in the bases sequence $b_1, \ldots, b_s$ is defined as $x_0, x_1, \ldots$ with

$$x_n = (\Phi_{b_1}(n), \ldots, \Phi_{b_s}(n)) \in I^s,\ \forall n \geq 0, \tag{14}$$

where $\Phi_b(n)$ is the radical inverse function. The radical inverse function in base b is defined as

$$\Phi_b(n) = \sum_{i=0}^{\infty} a_i(n) b^{-j-1},\ \forall n \geq 0, \tag{15}$$

where $a_i(n)$ is the ith digit in the digit expansion of n in base b.

4.1.4. Niederreiter-Xing Sequence

In [34] and [35] Niederreiter and Xing develop two methods for creating a digital (t, s)-sequence over F_b based on an algebraic function field with full constant field F_b, genus t, and containing at least $s+1$ rational places. Niederreiter-Xing sequence construction III is constructive, assuming that defining equations for the function field are given and that $s+1$ rational places are known. We use the implementation of construction 18 in [36].

4.1.5. Weyl Sequence

Weyl sequences are defined by

$$x_n = (\{n\theta_1\}, \{n\theta_2\}, \ldots, \{n\theta_s\}),\ \ n = 1, 2, 3, \ldots$$

where s is the dimension and $\{x\}$ is the fractional part of x. It is well known that a Weyl sequence is uniformly distributed iff θ_i are independent irrational numbers. Weyl sequences are used in different variants in literature. An important issue with respect to their quality in terms of uniformity of distribution is the amount or degree of irrationality of the employed starting vector $\Theta = (\theta_1, \ldots, \theta_s)$. Whereas deDoncker et al. [22] investigate Richtmyer rules where $\theta_i = \sqrt{p_i}$ with p_i being the i-th prime, we use a special type of Zinterhof sequences [37] where $\theta_i = e^{1/i}$:

$$x_n = (\{ne^{1/1}\}, \ldots, \{ne^{1/s}\}),\ \ n = 1, 2, 3, \ldots, \tag{16}$$

These sequences have been shown to exhibit excellent distribution behavior and they excel by their ease of construction and implementation even for non-specialists.

4.2. Test Functions

A widely used test function for numerical integration of improper integrals, which was first used by Sobol' in 1973 [38], is

$$t(x) = \frac{1}{x_1^{\alpha_1} \cdots x_s^{\alpha_s}}, \tag{17}$$

where s is the dimension and $0 < \alpha_i < 1,\ \forall i \in 1, \ldots, s$. We use a slightly less general version of this test function of the form

$$f(x) = \prod_{i=1}^{s} \frac{1}{x_i^{\alpha}}, \tag{18}$$

where s is the dimension and $0 < \alpha < 1$. For $f(x)$ clearly $\int_{(0,1)^s} f(x)dx = \left(\frac{1}{1-\alpha}\right)^s$ and the value of α determines the severity of the singularity, i.e. the gradient of the function (see Fig. 1). Additionally to be able to analyze situations where the singularity is inside the interval, as opposed to the origin, we can shift the function by a value $o > 0$ as follows

$$f_o(x) = \prod_{i=1}^{s} \frac{1}{\{x + o\}^{\alpha}}, \tag{19}$$

where $\{x\}$ is the fractional part of x (see Fig. 1).

Unless specified otherwise, we use dimension $s = 10$ and no shift in the test function. We evaluate up to $10000000 = 10^7$ integration nodes and set the bound B to 1000000.

4.3. Test Environment

The hardware platform used in our experiments is a classical heterogeneous cluster architecture. The machines are specified in Table 1.

The table contains information concerning the CPU of the machines, and in particular their architecture, speed and memory. The entries in column "stream" is the number of the leaped substream of the QMC sequence used on this machine.

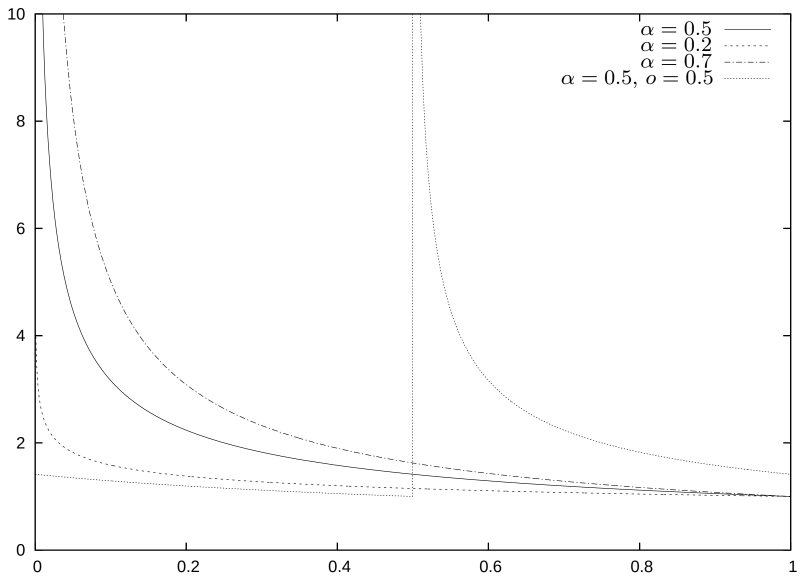


Fig. 1. Coordinate function of test function with various values of α and shift o as given.

Table 1. Specification of machines in the heterogenous network.

CPU(arch)	CPU(MHz)	Memory (kB)	stream	machine
AMD Athlon(tm) Processor	1244.719	513852	root	1
AMD Athlon(tm) MP 2800+	2133.468	2064584	stream0	2
AMD Athlon(tm) XP 2800+	2083.121	513852	stream1	3
AMD Athlon(tm) XP 2800+	2083.123	513852	stream2	4
AMD Athlon(tm) XP 2800+	2083.139	255308	stream3	5
AMD Athlon(tm) XP 2800+	2083.134	513852	stream4	6
AMD Athlon(tm) XP 2800+	2083.149	513852	stream5	7
AMD Athlon(tm) XP 2000+	1666.747	255308	stream6	8
AMD Athlon(tm) XP 2000+	1659.642	255308	stream7	9
AMD Athlon(tm) XP 2000+	1659.642	255308	stream8	10
AMD Athlon(tm) XP 2000+	1659.627	255308	stream9	11
AMD Athlon(tm) Processor	1244.732	513852	stream10	12

"Root" means that no actual integration is done but that this machine is the coordinator of the calculations. In a test on eight machines, machine one is the control node, machine two uses stream 0, machine three uses stream 1, and so on. When for a test the number of machines is stated this table gives information on which machines the streams run on and thus with which relative speed.

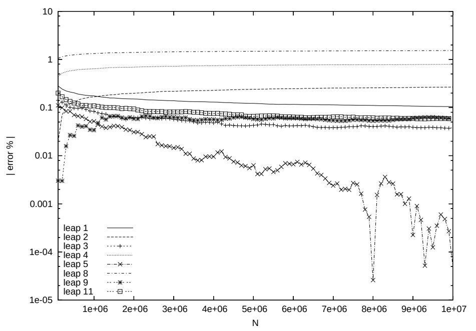
In order to simulate a more heterogeneous environment we artificially speed up or slow down single machines to simulate faster or slower machines contributing to the integration process. Due to the potential heterogeneity of GRID environments this speedup is greatly exaggerated; the corresponding machine running faster or slower does so by a factor of 10^3. Additionally, we also compare this extreme case of speedup to a more modest case, where the affected machine consumes twice as much (in case of acceleration) or half the amount (in case of slow down) of QMC points.

4.4. Single Substream Results

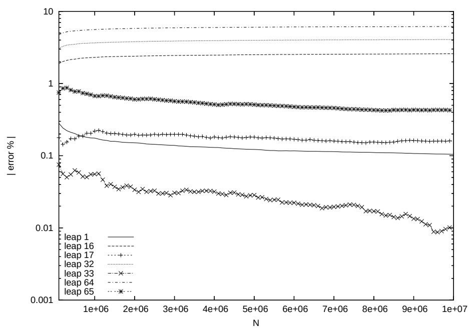
In this section we investigate the integration result accuracy when single leaped substreams are used for generating the integration nodes. The rationale for this is to assess the quality of these substreams – in case of unbalanced load substreams generated on faster PEs will contribute more integration nodes to the overall result than others and consequently their quality will propagate into the result to some extent.

First we focus onto the Sobol' sequence. We aim at showing that the behavior of the Sobol' sequence as documented by Schmid and Uhl [10] can also be seen when integrating functions with singularities. Note that for figures the ordinate gives the absolute value of the error percentage displayed in a logarithmic scale. We investigate the behavior of leaps of the form 2^i (depicted as lines) and $2^i + 1$ (depicted as lines with points), for $i = 1, \dots, 6$ (additionally leap 11 is shown for stream 0, see below). For a reference value we display the result of the original sequence (leap 1) with an offset equal to the stream number. Apart from using substreams with different leap values starting with the first point of the sequence ("stream 0"), we also investigate those substreams initialized with the second and third point, respectively ("stream 1" and "stream 2").

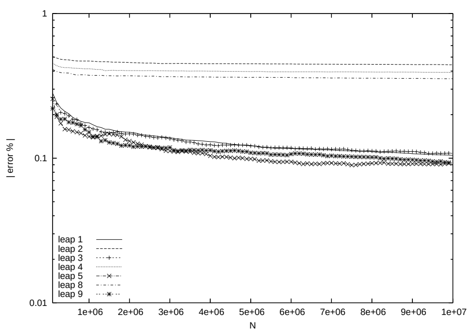
The general impression of the results for the Sobol' sequence in Fig. 2 confirms the findings of earlier work [9,10,11] where stream 0 of Sobol' and binary (t,s)-sequences has shown severe integration result degradation when using leaps of the form 2^i and a lower amount of degradation for leaps of the form $2^i + 1$, while still giving worse results as compared to the original sequence. However, there are also subtile differences shown in the results. To start with, in case of stream 2 and large leaps we notice that **all** results are superior to the result of the sequential sequence. Leaps of the form $2^i + 1$ do not only perform better as compared to those of the form 2^i but actually do improve the results of the original sequence in almost all cases considered (we notice an even significant improvement in single cases, e.g.



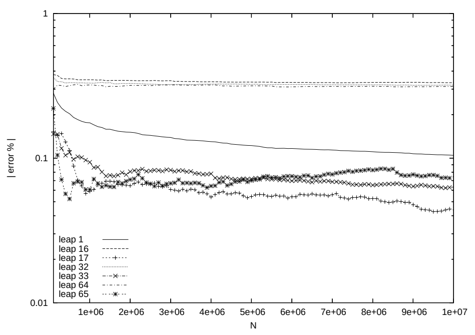
Stream 0, leaps 2–9 and 11



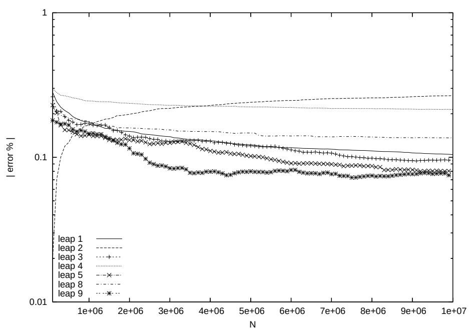
Stream 0, leaps 16–65



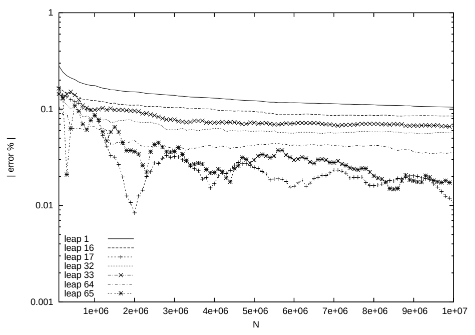
Stream 1, leaps 2–9



Stream 1, leaps 16–65



Stream 2, leaps 2–9



Stream 2, leaps 16–65

Fig. 2. Stream behavior for the Sobol' Sequence, streams zero to two.

stream 0 - leap 5 and leap 33). These effects (and differences to earlier findings) may be due to the fact that we use a test function with singularity located in the origin and that leaping in general reduces the share of points taken from the start of the original sequence (which is known to be of lower quality [39,40]). It is also interesting to note that there are significant differences among the results of the different streams. The overall trend suggests that result degradations caused by leaping are less severe for streams initialized at some distance from the start of the original sequence.

Fig. 3 shows the results of the identical experimental setup applied to Halton sequences. A very different behavior is displayed.

Clearly, the observation that leaps of the form $2^i + 1$ perform better than leaps of the form 2^i, as seen in the Sobol' sequence, doesn't hold for the Halton sequence at all. For streams 0 and 2 the leaps of the form 2^i are better, while for stream 1, with exception of 2^{17} and 2^{65}, the streams of the form $2^i + 1$ are better. Overall, leaping has a disastrous effect on the integration results which are severely degraded in almost all cases (only stream 1 - leap 33 and stream 2 - leaps 16,32,64 improve the result of the original sequence). Note that again the results differ a lot among the different streams, especially for larger leaps.

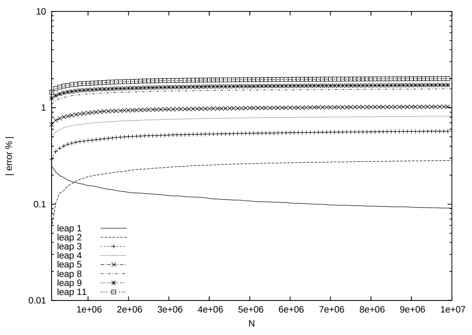
Fig. 4 applies the test scenario to the Niederreiter-Xing sequence. Again, significantly different behavior with respect to splitting sensitivity may be observed.

An almost opposite behavior to the Halton sequence is found for the Niederreiter-Xing sequence. All leap forms considered improve the results of the original sequence and for larger leaps the improvement is seen to a higher extent. There is no clear relation between leaps of the form 2^i and 2^i+1 as seen for Sobol' or Halton sequences.

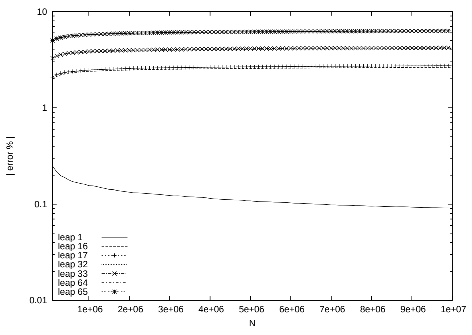
For Faure and Zinterhof sequences we restrict our investigations to stream 0. Fig. 5 shows that also for the Faure sequence result degradation occurs but less frequent and on an irregular basis (only leaps 5 and 32 exhibit worse results as compared to the original sequence); in most cases we note a moderate improvement. This is somewhat surprising since the Faure sequence is a very specialized form of Niederreiter-Xing sequences and was expected to behave similarly. This result suggests that also the Niederreiter-Xing sequences might exhibit degraded results under leaping for some specifically selected parameters.

Finally Fig. 6 shows the results of leaping applied to the Zinterhof sequence. Similar to Niederreiter-Xing sequences, we do not find result degradations but improvements as a result of leaping and there is no systematic and significant difference among leaps of different form.

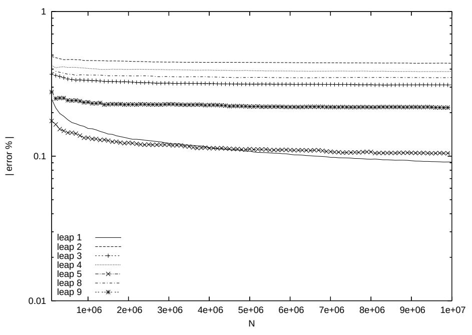
We have seen that the different types of QMC sequences react differently to the use of single substreams. Whereas Niederreiter-Xing and Zinterhof sequences turn out to be very stable and give even slightly better results than the original sequences, Sobol', Halton, and Faure sequences show degraded integration results for some settings. Whereas the effects are somewhat structured and occur frequently in the case of the Sobol' sequence, the contrary is true for the Faure sequence. The Halton



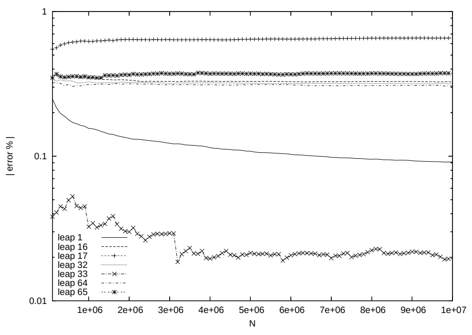
Stream 0, leaps 2–9 and 11



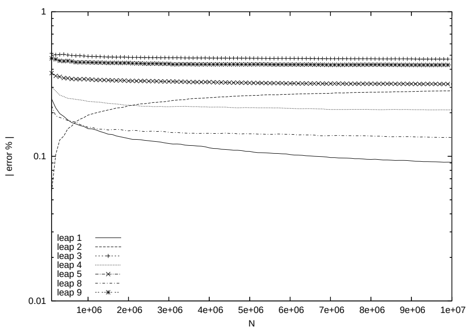
Stream 0, leaps 16–65



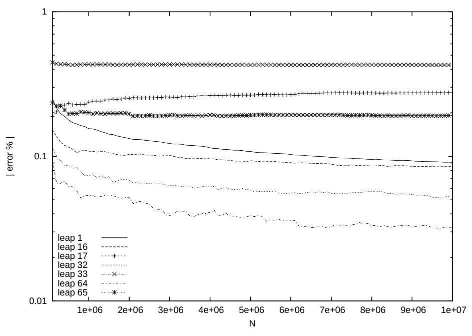
Stream 1, leaps 2–9



Stream 1, leaps 16–65

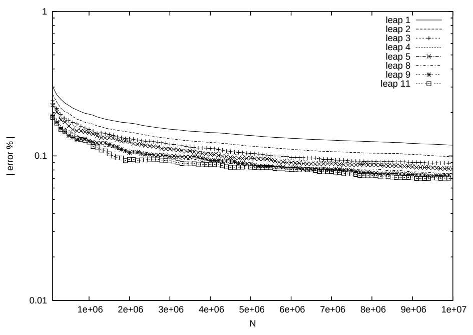


Stream 2, leaps 2–9

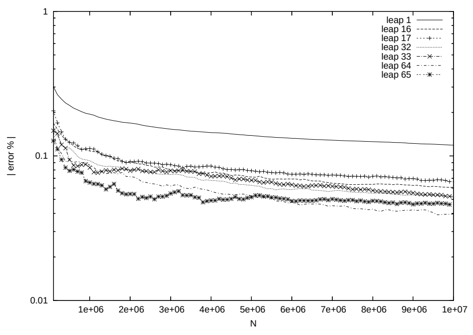


Stream 2, leaps 16–65

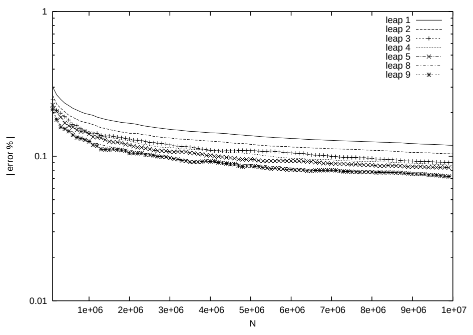
Fig. 3. Stream behavior for the Halton Sequence, streams zero to two.



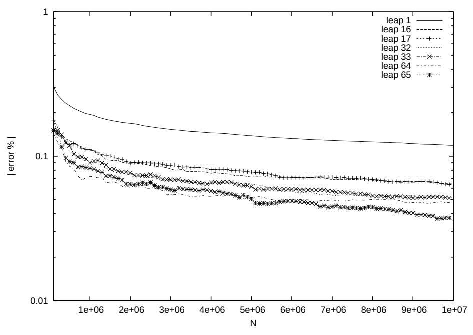
Stream 0, leaps 2–9 and 11



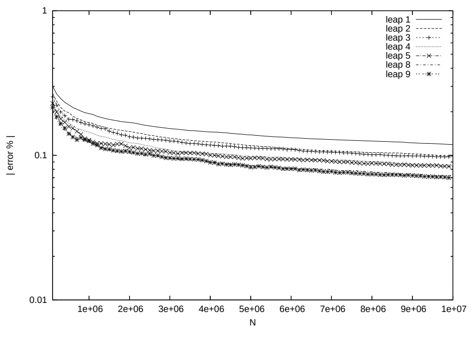
Stream 0, leaps 16–65



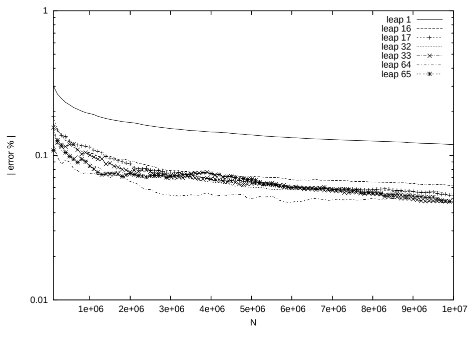
Stream 1, leaps 2–9



Stream 1, leaps 16–65

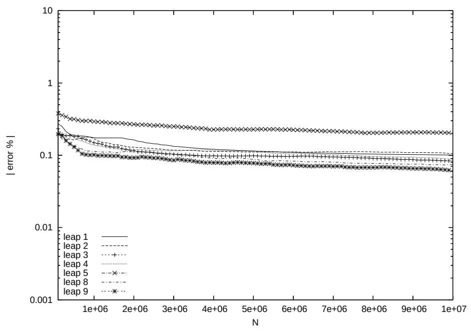


Stream 2, leaps 2–9

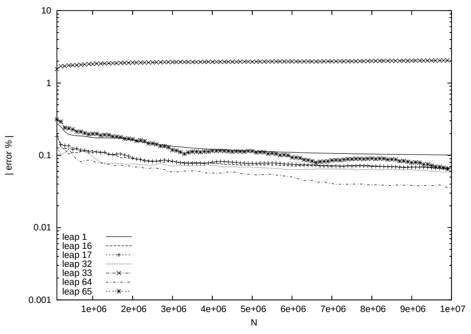


Stream 2, leaps 16–65

Fig. 4. Stream behavior for the Niederreiter-Xing Sequence, streams zero to two.

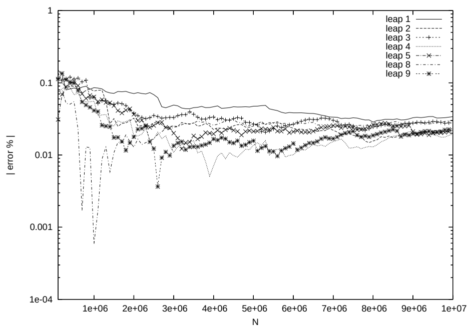


Stream 0, leaps 2–9

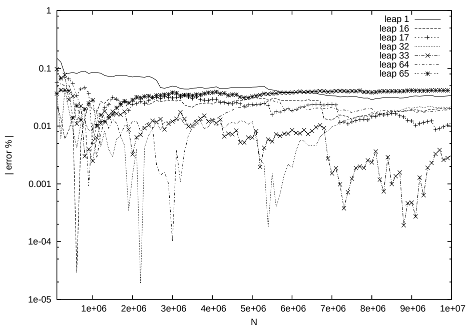


Stream 0, leaps 16–65

Fig. 5. Stream zero behavior for the Faure Sequence.



Stream 0, leaps 2–9



Stream 0, leaps 16–65

Fig. 6. Stream zero behavior for the Zinterhof sequence.

sequence is least suited for leaping and delivers almost consistently significantly worse results as compared to the original. We have also observed that different initialization (different streams) but equal leap form also may lead to significantly different behavior.

Furthermore the behavior of the leaped substreams when used for integrating improper integrals depends also on the position of the singularity. When shifting the singularity from the origin into the unit interval (e.g. in the following example a shift of 0.5 – see Fig. 1), the results differ significantly to the original case. Fig. 7 shows the results for stream 2 and large leaps of the Halton sequence (see lower right graph of Fig. 3 for the original test function).

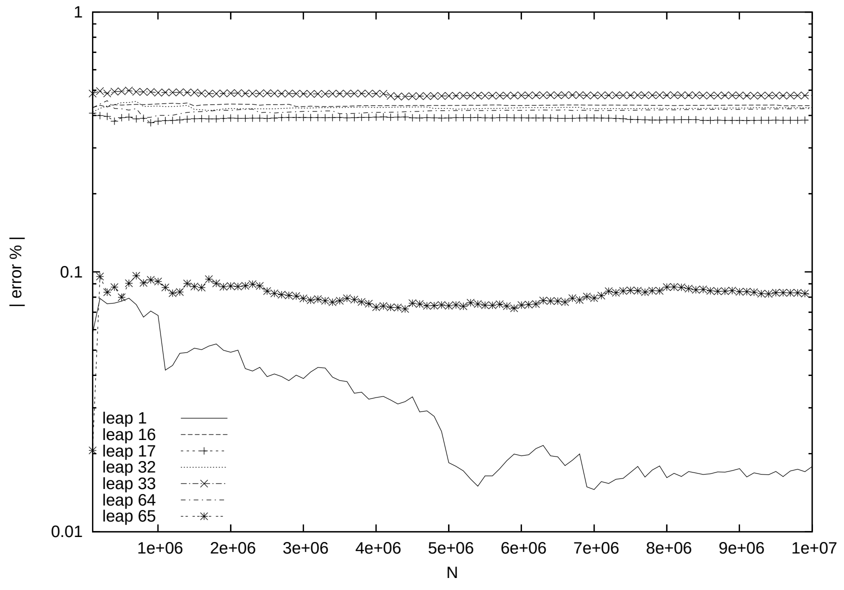


Fig. 7. Stream two, leaps one and 16–65 of the Halton sequence using the function $f_{0.5}(x)$.

Whereas 2^i leaps improve the Halton sequence results for the original test function, all forms of leaps degrade the results for the shifted version. A possible explanation of this effect is that most QMC sequences avoid the origin in an L-shaped or hyperbolic shaped region [41,42]. Therefore we are dealing with the more benevolent case when the singularity is situated in the origin. Thus in the general case one can expect further deterioration of the stream when the singularity is moved into the interval.

In the following section, we will investigate the possible impact of the behavior of single substreams on integration results when leaping based QMC point set distribution is used in GRID environments.

4.5. Multiple Substream Results

First let us consider the execution efficiency in a distributed environment of the splitting (leap frogging) and blocking based distribution approach as compared with a synchronized and centralized technique (denoted client/server). In the client/server version we have one PE which generates and distributes the QMC integration points and collects the results (control element CE), and a number of PEs where $f(x)$ is calculated. For a high number of PEs this model becomes a problem since the only source for new points is the server which can only generate points at a fixed rate. Furthermore, sending the points and results over a network generates a high amount of traffic, thus it is necessary to shift the generation of points from the server to the client. The test is conducted on a heterogenous network consisting of 12 computers (for machine specification see Section 4.3). One machine is always setup as CE while the rest are PEs. For leaping the leap size was set to 11 (i.e. the number of PEs) and the block size for the blocking mode was set to 500. The average runtime of five tests was used to calculate the speedup as shown in Table 2. We see that for both blocking and leaping the speedup as compared to the client/server version is clearly in a range where splitting can be considered useful.

Table 2. Speedup from splitting.

Mode	Speedup	Average Runtime (milliseconds)
client/server	1	6092.2
blocking	3.12	1952.6
splitting	3.27	1861.8

The reason for leaping being slightly better than blocking stems from the fact that in the leaping mode only control messages need to be exchanged while the blocking variant periodically has to request new blocks of QMC nodes. As we have discussed earlier, there is also the possibility to bring the blocking mode closer to leaping by using large blocks.

Now we turn to results concerning integration accuracy. To start we again use 11 PEs and consequently a leap factor of 11. Fig. 8 displays the results for the Sobol' sequence. The client/server integration result in the figure serves as the reference value for all subsequent tests as well. For leaping, we use a standard non-synchronized leaping approach (denoted as “leap” in the plots) and the labels “one-fast” and “one-slow” denote the leaping scenarios with one artificially slowed or speed up PE by a factor of 10^3 as described in Section 4.3 (applied to stream 0 in either case).

We see that slowing down one PE significantly and employing leaping without precautions on systems with moderate heterogeneity does not change the integration result in this case. However, when speeding up one PE the integration result is improved. This result corresponds well to the results of single substream investigations where stream 0 of leap 11 also shows better behavior as compared to the

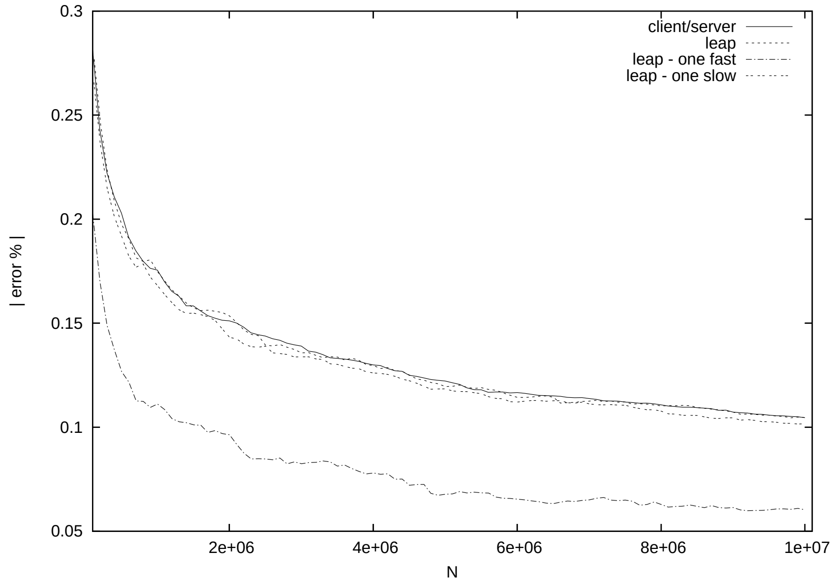


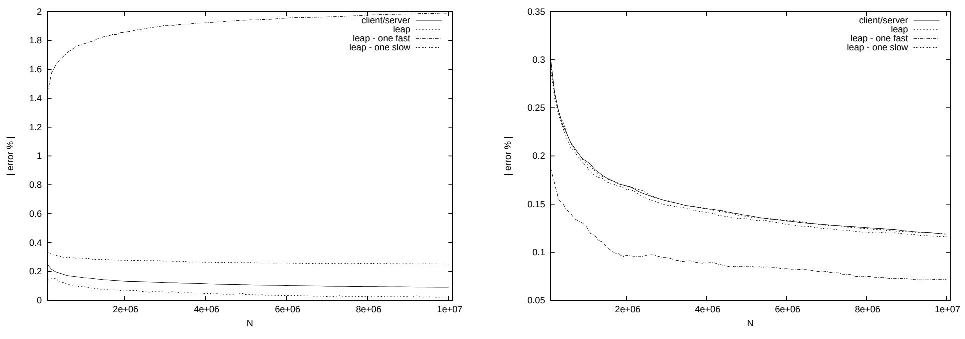
Fig. 8. Comparison of leaping for the Sobol' sequence utilizing 12 machines.

original sequence (compare Fig. 2 top left graph). Note that speeding up one node by a factor of 10^3 means that the 10 slower nodes only contribute 1% of all QMC integration nodes which explains the propagation of the single substream behavior into the final result. Slowing down one node by the same amount (which in fact means that one stream is missing almost entirely) on the other hand obviously does not affect the result at all.

Fig. 9 (left plot) shows the results of the same setting applied to the Halton sequence. We again observe a significant deviation from the client/server result only in case of speeding up one PE, and the deviation is quite severe. This behavior relates to the single substream behavior which shows the same for the single stream 0 (compare Fig. 3 top left graph).

Also in the case of the Niederreiter-Xing sequence the single substream behavior of stream 0 of leap 11 (see Fig. 4 top left graph) is propagated into the result of the distributed execution with one speed up PE whereas the two other leaping variants are not affected (Fig. 9 (right plot)).

As a consequence of these results, changing the leap factor is expected to potentially change the overall result significantly. This is confirmed in Fig. 10 where we use 8 PEs (resulting in a leap factor 8) and the Sobol' sequence. Again in perfect accordance to the results seen before speeding up stream 0 affects the integration result, in this case a severe degradation is observed (as it is expected from a single



Halton

Niederreiter-Xing

Fig. 9. Comparison of leaping for the Halton and Niederreiter-Xing sequences utilizing 12 machines.

substream behavior). However, when speeding up stream 0 by a factor of 2 only (denoted as "leap-one fast-double") the integration result is even improved as compared to the client/server reference which does not at all correspond to the former result. Slowing down one PE by a factor of 10^3 results in some degradation in this setting which is also true (but with minor significance) for slowing down by a factor of 2.

When we change the number of machines employed to 10 machines (9 PEs, resulting in leap factor 9), we note in Fig. 11 that both types of speeding up stream zero improve the result. Whereas this is to be expected in principle based on the single substream results, the amount of improvement when speeding up by a factor of 10^3 exceeds the improvement of single substream execution.

Finally we investigate the influence of which stream is affected by a slow down or speed up in distributed execution. Using again 8 PEs but speeding up stream 8 in this case (as opposed to stream 0 as shown in Fig. 10) is shown to deliver degraded results as well in the case of the Sobol' sequence (Fig. 12), however, the degradation is much less severe in this case. The Niederreiter-Xing sequence does exhibit result improvement again. Note also that for the client/server result the Sobol' sequence is superior to the Niederreiter-Xing sequence.

The results of leaping applied in our experimental setting may be summarized and interpreted as follows:

- Properties of the single substreams are propagated to integration results using multiple substreams in distributed execution to some extent.
- Systems exhibiting moderate heterogeneity or systems with a single failing PE are not severely affected by this phenomenon.
- Systems with a large variety in processing speed may produce very unreliable

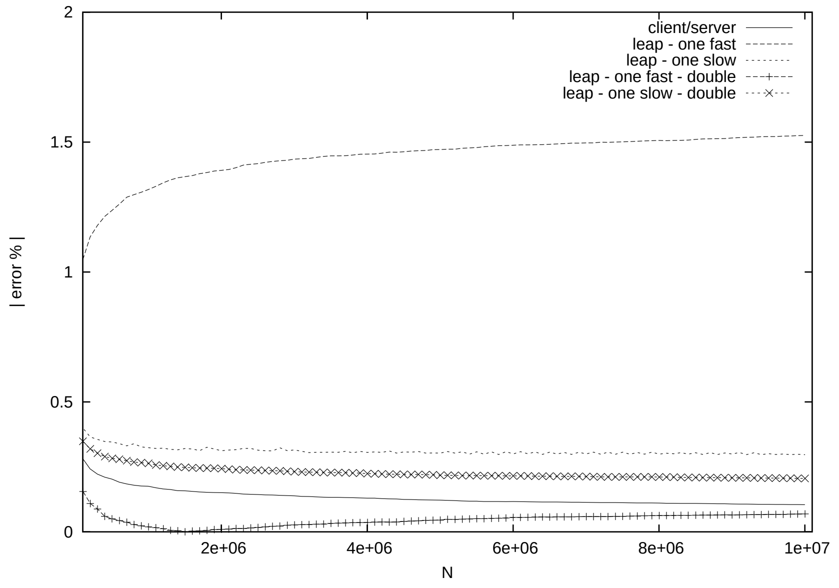


Fig. 10. Influence of artificial speedup on the results in the case of the Sobol' sequence using nine machines.

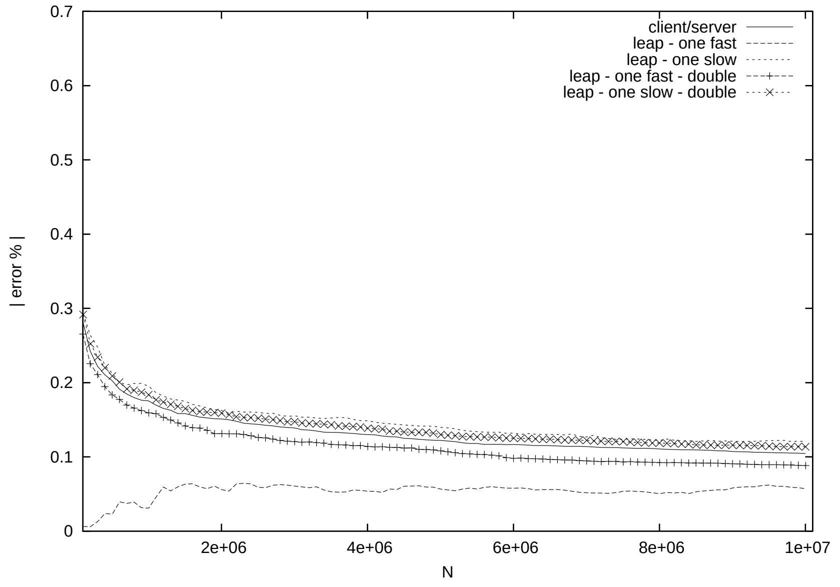


Fig. 11. Influence of artificial speedup on the results in the case of the Sobol' sequence using ten machines.

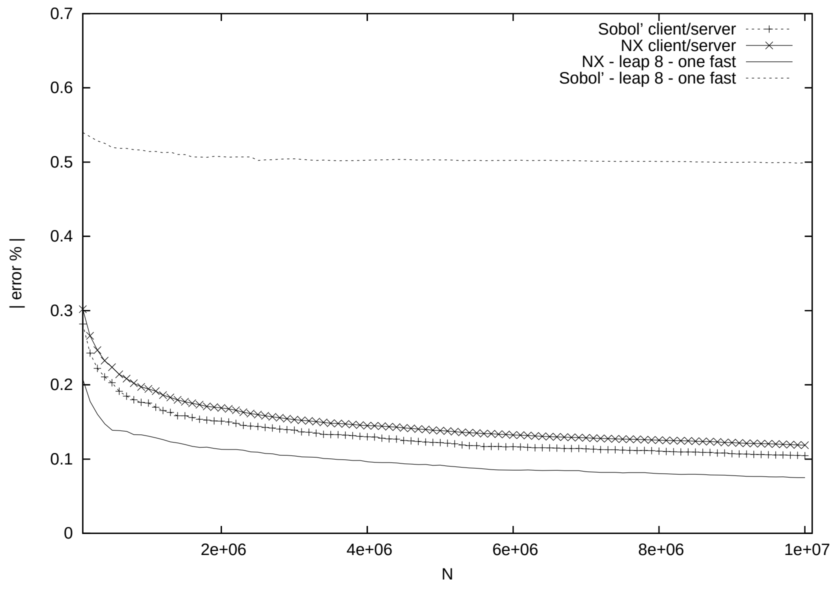


Fig. 12. Comparison of leaping for the Sobol' and Niederreiter-Xing sequences utilizing nine machines and modifying stream 8.

and poor results when leaping is applied. In most cases these effects can be predicted by analyzing single substream behavior, but not all numerical effects may be explained in this manner sufficiently.

When turning to GRID environments this means that due to the potential extreme heterogeneity leaping can not be used without extreme care in such environments. While leaping turns out to be able to handle single failing PEs well (the case of "one-slow" in the results), the sensitivity towards different processing speeds is of course a significant problem when QMC node sets with low quality leaped substreams are used. In this case only investigation of single substream behavior and corresponding parameter selection (e.g. choosing leap factors not equal to the number of participating PEs) **or** synchronization among PEs helps out. Since both approaches do not really contribute to facilitate the use of leaping in an efficient and transparent way, other strategies should be used under such conditions.

5. Conclusion and Future Work

Leaped substreams of different types of QMC point sets have turned out to behave quite differently. Whereas Sobol', Halton, and Faure sequences exhibit single leaped substreams with extremely low quality, Niederreiter-Xing and Zinterhof sequences are shown to behave very stable under splitting in our experiments.

On systems with a large amount of heterogeneity in terms of processing speed, the properties of single substreams may be reflected in the final result (which is of course a problem in case of substreams with extremely low quality like Halton, Sobol', and Faure sequences). This effect does not occur in systems with moderate heterogeneity and in systems with single failing PEs.

Consequently, using leaped substreams of QMC sequences on single PEs as a general strategy in GRID environments can not be recommended. Our results however indicate that certain types of sequences may be employed in this scenario without taking specific precautions (Niederreiter-Xing and Zinterhof sequences) and we have demonstrated that a moderate amount of heterogeneity does not lead to severe result degradation, even in scenarios where low quality substreams are used.

In future work we will focus on theoretical results backing up the experimental findings, especially with respect to the extreme differences in splitting sensitivity among the different QMC point sets. Additionally we will investigate a leaped substream scenario being able to handle additional computing resources, using leap factors larger as the number of PEs starting the computation thereby allowing PEs to be added.

Acknowledgments

The work described in this paper is partially supported by the Austrian Grid Project, funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ 4003/2-VI/4c/2004.

References

[1] E. deDoncker, R. Zanny, and K. Kaugars. Distributed numerical integration algorithms and applications. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics, and Informatics (SCI'00)*, pages 244–249, 2000.

[2] A.R. Krommer and C.W. Überhuber. *Numerical Integration on Advanced Computer Systems*, volume 848 of *Lecture Notes in Computer Science*. Springer, Berlin, 1994.

[3] A.R. Krommer and C.W. Überhuber. *Computational integration*. SIAM, Philadelphia, 1998.

[4] R. Schürer and A. Uhl. An evaluation of adaptive numerical integration algorithms on parallel systems. *Parallel Algorithms and Applications*, 18(1–2):13–26, 2003.

[5] G. Evans. *Practical numerical integration*. Wiley, Chichester, 1993.

[6] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Number 63 in CBMS-NSF Series in Applied Mathematics. SIAM, Philadelphia, 1992.

[7] K. Entacher, P. Hellekalek, and P. L'Ecuyer. Quasi-Monte Carlo node sets from linear congruential generators. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 188–198. Springer, 2000.

[8] S. Li, K. Kaugars, and E. deDoncker. Grid-based numerical integration and visualization. In *Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05)*, pages 260–265. IEEE Computer Society Press, 2005.

[9] W. Ch. Schmid and A. Uhl. Parallel quasi-Monte Carlo integration using (t,s)-

sequences. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Parallel Computation. Proceedings of ACPC'99*, volume 1557 of *Lecture Notes on Computer Science*, pages 96–106. Springer-Verlag, 1999.

[10] W. Ch. Schmid and A. Uhl. Techniques for parallel quasi-Monte Carlo integration with digital sequences and associated problems. *Mathematics and Computers in Simulation*, 55:249–257, 2001.

[11] K. Entacher, T. Schell, W. Ch. Schmid, and A. Uhl. Defects in parallel Monte Carlo and quasi-Monte Carlo integration using the leap-frog technique. *Parallel Algorithms and Applications*, 18(1–2):27–47, 2003.

[12] M. Mascagni and A. Karaivanova. A parallel quasi-Monte Carlo method for solving systems of linear equations. In P. Sloot et al., editors, *The 2002 International Conference on Computational Science - ICCS 2002*, volume 2330 of *Lecture Notes in Computer Science*, pages 598–608. Springer Verlag, Berlin, Germany, May 2002.

[13] M. Mascagni and A. Karaivanova. A parallel quasi-Monte Carlo method for computing extremal eigenvalues. In K. T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 369–380. Springer-Verlag, 2002.

[14] V. Alexandrov, E. Atanassov, and I. Dimov. Parallel quasi Monte Carlo methods for linear algebra problems. *Monte Carlo Methods and Applications*, 10(3-4):213–219, 2004.

[15] Yaohang Li and Michael Mascagni. Grid-based Quasi-Monte Carlo applications. *Monte Carlo Methods and Appl.*, 11(1):39–55, 2005.

[16] J.W.L. Wan, K. Lai, A.W. Kolkiewicz, and K.S. Tan. A parallel quasi Monte Carlo approach to pricing multidimensional americal options. *International Journal of High Performance Computing and Networking*, 2006. To appear.

[17] R. Schürer. Parallel high-dimensional integration: quasi-Monte Carlo versus adaptive cubature rules. In V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, editors, *The 2001 International Conference on Computational Science - ICCS 2001*, volume 2073 of *Lecture Notes in Computer Science*, pages 1262–1271, San Francisco, CA, USA, May 2001. Springer Verlag, Berlin, Germany.

[18] B.C. Bromley. Quasirandom number generators for parallel Monte Carlo algorithms. *Journal of Parallel and Distributed Computing*, **38**:101–104, 1996.

[19] J.X. Li and G.L. Mullen. Parallel computing of a quasi-Monte Carlo algorithm for valuing derivatives. *Parallel Computing*, 26(5):641–653, 2000.

[20] E. deDoncker, A. Genz, and M. Ciobanu. Parallel compuation of multivariate normal probabilities. *Computing Science and Statistics*, 30, 1999.

[21] E. deDoncker, R. Zanny, M. Ciobanu, and Y. Guan. Distributed quasi-Monte Carlo methods in a heterogeneous environment. In *Proceedings of the Heterogeneous Computing Workshop 2000 (HCW'2000)*, pages 200–206. IEEE Computer Society Press, 2000.

[22] E. deDoncker, R. Zanny, M. Ciobanu, and Y. Guan. Asynchronous quasi-Monte Carlo methods. In *Proceedings of the High Performance Computing Symposium 2000 (HPC'00)*, pages 130–135, 2000.

[23] L. Cucos and E. deDoncker. Distributed QMC algorithms: new strategies for and performance evaluation. In *Proceedings of the High Performance Computing Symposium 2002 (HPC'02)/Advanced Simulation Techniques Conference*, pages 155–159, 2002.

[24] G. Ökten and A. Srivivasan. Parallel quasi-Monte Carlo methods on a heterogeneous cluster. In K. T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 406–421. Springer-Verlag, 2002.

[25] A. Srinivasan. Parallel and distributed computing issues in pricing financial derivatives through quasi-Monte Carlo. In *Proceedings of the International Parallel & Distributed Processing Symposium 2002 (IPDPS'02)*, Fort Lauderdale, FL, USA, April 2002. IEEE Computer Society Press.

[26] E. deDoncker and Y. Guan. Error bounds for the integration of singular functions using equidistributed sequences. *Journal of Complexity*, 19:259–271, 2003.

[27] Peter Zinterhof. High dimensional improper integration procedures. In Civil Engineering Faculty Technical University of Košice, editor, *Proceedings of Contributions of the 7th International Scientific Conference*, pages 109–115, Hroncova 5, 04001 Košice, SLOVAKIA, May 2002. TULIP.

[28] Rudolf Schürer and Wolfgang Ch. Schmid. Mint - the database of optimal (t,m,s)-net parameters. online: `http://mint.sbg.ac.at/`.

[29] Ilya M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[30] Harald Niederreiter. Point sets and sequences with small discrepancy. *Monatshefte für Mathematik*, 104:273–337, 1987.

[31] Gary L. Mullen, Arijit Mahalanabis, and Harald Niederreiter. Tables of (t, m, s)-net and (t, s)-sequence parameters. In Harald Niederreiter and P. J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume Lecture Notes in Statistics of *106*, pages 58–86. Springer-Verlag, 1995.

[32] Henry Faure. Discrépance de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, 41:337–351, 1982.

[33] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimension integrals. *Numer. Math.*, 2:84–90, 1960. Berichtigung, ibid., (1960), p. 196.

[34] Harald Niederreiter and Chaoping Xing. Low-discrepancy sequences and global function fields with many rational places. *Finite Fields and Their Applications*, 2:241–273, 1996.

[35] Chaoping Xing and Harald Niederreiter. A construction of low-discrepancy sequences using global function fields. *Acta Arithmetica*, 71(1):87–102, 1995.

[36] Andrew T. Clayman, K. Mark Lawrence, Gary L. Mullen, Harald Niederreiter, and N. J. A. Sloane. Updated tables of parameters of (t, m, s)-nets. *Journal of Combinatorial Designs*, 7:381–393, 1999.

[37] Peter Zinterhof. Einige zahlentheoretische methoden zur numerischen Quadratur und Interpolation. *Sitzungsberichte der Österreichischen Akademie der Wissenschaften, math.-nat.wiss. Klasse Abt. II*, 177:51–77, 1969.

[38] I.M. Sobol'. Calculation of improper integrals using uniformly distributed sequences. *Soviet Math Dokl.*, 14(3):734–738, July 1973.

[39] L. Kocis and W.J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software*, 23(2):266–294, 1997.

[40] I. Radović, I.M. Sobol, and R.F. Tichy. Quasi-monte carlo methods for numerical integration: Comparison of different low discrepancy sequences. *Monte Carlo Methods and Appl.*, **2**(1):1–14, 1996.

[41] Jürgen Hartinger, Reinhold Kaindorfer, and Volker Ziegler. On the corner avoidance properties of various low-discrepancy sequences. 2004. submitted.

[42] Art B. Owen. Halton sequences avoid the origin. Technical report, Stanford University, 2004.