Accelerating Lattice Chromodynamics (LQCD) Simulations with Value Prediction

Jie Tang, Shaoshan Liu, Chen Liu, Christine Eisenbeis, Jean-Luc Gaudiot

Accelerating Lattice Chromodynamics (LQCD) Simulations with Value Prediction

Jie Tang[1], Shaoshan Liu[2], Chen Liu[3], Christine Eisenbeis[4], Jean-Luc Gaudiot[5]

[1] South China University of Technology, China
[2] PerceptIn
[3] Clarkson University
[4] INRIA, France
[5] University of California, Irvine
{cstangjie@scut.edu.cn, shaoshan.liu@perceptin.io, cliu@clarkson.edu,
christine.eisenbeis@inria.fr, gaudiot@uci.edu}

Abstract. Communication latency problems are universal and have become a major performance bottleneck as we scale in distributed computing architectures. Specifically, research institutes around the world have built specialized supercomputers with powerful computation units in order to accelerate scientific computation. However, the problem often comes from the communication side instead of the computation side. In this paper we first demonstrate the severity of communication latency problems. Then we use Lattice Quantum Chromo Dynamic (LQCD) simulations as a case study to show how value prediction techniques can reduce the communication overheads, thus leading to higher performance without adding more expensive hardware. In detail, we first implement a software value predictor on LQCD simulations: our results indicate that 22.15% of the predictions result in performance gain and only 2.65% of the predictions lead to rollbacks. Next we explore the hardware value predictor design, which results in a 20-fold reduction of the prediction latency. In addition, based on the observation that the full range of floating point accuracy may not be always needed, we propose and implement an initial design of the tolerance value predictor: as the tolerance range increases, the prediction accuracy also increases dramatically.

1 Introduction

With the rise of new applications such as deep learning and distributed query systems, Big Data Infrastructure and Many-Core Architectures have become the mainstream in the industry as well as the academia. In conventional superscalar architectures, the computation latency dominates the sequential sections. In many-core architectures, however, the communication latency increases with the number of cores and thus it takes a large portion of the execution time [1, 2]. In high performance computing environments, such as large-scale scientific simulations, the communication latency can even reach thousands of cycles [3]. Note that in this paper, we follow the convention in [4] and define value communication latency as the time to satisfy any true data dependence between threads. It includes transmission delays, which depend

on the communication network, and the blocking latency, which is the time the consumer core idles to wait for the result to be produced.

To address the communication problem, value prediction techniques may allow the "parallelization" of the sequential portion by predicting values before they are produced. With value prediction, the consumer core or node can predict the requested value before it is produced and proceed with its execution. This means that, when the value is successfully predicted, the communication overhead is eliminated and both the consumer and the producer can execute in parallel. In this paper, we use the Lattice Quantum Chromo Dynamic (LQCD) simulation as a case study to evaluate how value prediction techniques can reduce the communication overhead of large-scale scientific simulations.

Quantum Chromo Dynamics (QCD) is the theory of the strong nuclear force that binds the constituents of sub-nuclear matter, including quarks and gluons, to form stable nuclei. Being able to calculate quantities in QCD is essential in advancing the cosmological models as well as the energy research. LQCD is a challenging computational field employing large scale numerical calculations to extract predictions of the QCD theory. LQCD is a well-established non-perturbative numerical method that is formulated on a grid or lattice of points in space and time.

We choose this particular application for two reasons: first, LQCD is a critical application that is being studied by researchers all over the world. Many supercomputers have been built to perform LQCD simulations [14]. As pointed out in [5], every time there is a serious increase in computational speed coming from a massively parallel supercomputer, LQCD breaks new ground in the understanding of sub-nuclear matter. Second, LQCD simulations involve intensive communication such that each compute node needs to communicate with its neighbors frequently. As a result, communication has become the major performance bottleneck in LQCD simulations. Due to its special communication behavior, LQCD has also been proposed as a possible latency benchmark for high-performance computing [5].

The contributions of this paper are three-fold: first, using LQCD as a case study, we evaluate how value prediction techniques can address the communication latency issues. Second, we evaluate the software and hardware overhead of value predictor implementations. Last but not least, based on the value behaviors of LQCD simulations, we propose an initial design of the tolerance value predictor, which allows a small discrepancy between the predicted value and the actual value. The rest of the paper is organized as follows: in section 2, we review related work in value prediction and communication. In section 3, we illustrate the problem of communication in LQCD simulations. In section 4, we apply and evaluate value prediction techniques on the communication values in LQCD simulations. In section 5, we study the design and implementation of hardware value predictors to further reduce communication latency. In section 6, we present the initial design of the tolerance value predictor. Finally, conclusions are drawn in section 7.

2 Background

In this section, we introduce the background of LQCD simulations and review related work in value prediction and communication.

2.1 LQCD Simulations

In theoretical physics, QCD is a theory describing the interactions of the quarks and gluons making up hadrons (such as the proton, neutron or pion). Analytic or perturbative solutions in QCD are hard or impossible due to the highly nonlinear nature of the strong force. On the other hand, Lattice QCD (LQCD) is a well-established non-perturbative approach to solving the QCD theory of quarks and gluons. In detail, LQCD is a lattice gauge theory formulated on a grid or lattice of points in space and time. It provides a framework for investigation of non-perturbative phenomena such as confinement and quark-gluon plasma formation, which are intractable by means of analytic field theories. In LQCD, fields representing quarks are defined at lattice sites while the gluon fields are defined on the links connecting neighboring sites. This approximation approaches continuum QCD as the spacing between lattice sites is reduced to zero. Numerical lattice QCD calculations using Monte Carlo methods can be extremely computationally intensive, requiring the use of the largest available supercomputers. These simulations typically utilize algorithms based upon molecular dynamics or micro-canonical ensemble algorithms.

LQCD simulations are both computation and communication intensive. Figure 1 illustrates a 2D LQCD simulation grid: each node in the grid represents a point in space and time of the simulation, and it is called a *Spinor*. The spinors calculate the values of the current point, and at the end of each round of computation, each spinor communicates with all of its neighbors for data exchange. If two neighboring spinors are mapped to the same physical compute node, then the communications between them can happen through shared-memory with fairly low cost. However, if they are mapped to different physical compute nodes, then the communications have to go through the network with high cost. In Figure 1, the upper part of the grid gets mapped to one compute node and the lower part gets mapped to another compute node, thus the communications between the upper and lower parts of the grid need to use the network and incur very high performance overheads.

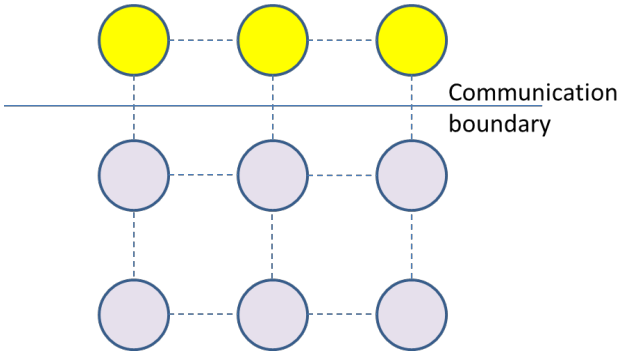

Fig. 1. Two dimensional LQCD simulation grid: each node is called a *spinor*

Due to its extremely high computing demand, LQCD has also been used as a benchmark for high-performance computing, an approach originally developed in the context of the IBM Blue Gene supercomputer [28]. Also, several supercomputers have been designed to solve LQCD problems [24, 25, 26, 27]. In detail, APENEXT [24] is the latest APE collaboration's series of parallel computers for computationally intensive calculations such as LQCD. The apeNEXT supercomputer consists of application-specific

accelerators for LQCD simulations. In [25], Piccoli *et al* discuss the application of workflow management systems to LQCD. In [26], Vranas *et al* describe the special programming requirements and the optimal supercomputer hardware architectures for LQCD. They argue that the IBM BG/L architecture is very well suited for LQCD studies. This suitability arises from the fact that LQCD is a regular lattice discretization of space into lattice sites, while the BG/L supercomputer is a discretization of space into compute nodes. In addition, they demonstrate a computational speedup of LQCD using up to 131,072 CPUs on the largest BG/L supercomputer available in 2007. As the number of CPUs is increased, the speedup increases linearly with sustained performance of about 20% of the maximum possible hardware speed. To address the communication problem in LQCD simulations, Ammendola *et al* present APENET, a high-speed, low-latency, 3D interconnect architecture optimized for LQCD simulations [27]. Although this design is able reach a peak bandwidth of 676 MB/s each direction and latency less than 10 us, communication still remains as the major performance bottleneck in LQCD simulations.

2.2 Value Prediction and Communication

Data dependencies are directly related to the degree of Instruction-Level Parallelism (ILP). This has long been regarded as the fundamental limiting factor to the parallelization of programs, as demonstrated by Lipasti and Chen [6]. To overcome this problem, several authors [6, 7, 8, and 9] have proposed value prediction techniques to exploit ILP through speculation. It has been demonstrated by Sazeides and Smith [7] that different value prediction mechanisms (last value predictors, stride predictors, finite context predictors) can achieve prediction accuracies that range from 56% to 92% and result in performance gains ranging from 7% to 23%. Also, Tullsen and Seng [10] proposed the register value prediction technique, which predicts that an instruction will produce the value that is already stored in the destination register; they demonstrated that this technique led to a 12% performance improvement. Mendelson and Gabbay [11] proposed the stride value predictor and compared its performance and potential to that of the last value predictor.

The aforementioned value prediction techniques were proposed in the context of superscalar processors. Several researchers [12, 13] have also proposed incorporating value prediction techniques in the context of thread-level speculation. For instance, Marcuello *et al.* [12] evaluated the potential for value prediction in a Clustered Speculative Multithreaded Architecture when speculating at the thread-level on the innermost loops of SPECint95 benchmarks. Oplinger *et al.* [13] evaluated the potential benefits to Thread-Level Speculation of memory, register, and procedure return value prediction.

With the advent of many-core architectures, the On-Chip Interconnection Networks (OCIN) technology has been proposed to handle the communication between processing elements [2]. Nonetheless, the latency of these networks is still too large and often leads to overall performance degradation. To address this problem, Steffan *et al.* [4] proposed to improve value communication for thread-level speculation through speculation, synchronization, and value prediction techniques. Also, Parcerisa *et al.* [15] proposed to reduce wire delay penalty through value prediction.

We believe that reducing communication latency through value prediction is a

viable technique. The core idea of this technique is to reduce communication latency by predicting the values to be communicated before these values are actually produced. Generally, when a data dependency exists between two nodes across the communication network, the consumer cannot move on with its computation until it has received the value from the producer. With value prediction, when the consumer is waiting for the value, it can predict the incoming value and moves on with its computation with the predicted value. If the predictions were successful, then performance gain can be generated. Otherwise, the consumer needs to rollback and restart the computation with the correct value.

The effectiveness of value prediction techniques depend on the predictabilities of the data values as well as the performance of value predictors. To understand the predictability of data values, we have previously incorporated information theory measures into Amdahl's formulation to model the data redundancy inherent to each program [16, 22]. Information theory [23] aims at the quantification of information. It was originally developed to identify the fundamental limits on the compressing and communicating of data in the presence of random noise. The key measure in information theory is information entropy, which quantifies the uncertainty (or randomness) of a random process. In our approach, we treat the data streams (*e.g.,* data production in each instruction) of programs as random processes and quantify the redundancy of data values in each data stream. This approach allows the identification of the potential of value prediction techniques. In this paper, we apply the theoretical model presented in our previous work [16, 22] to characterize the value predictability in high performance scientific simulations exemplified by LQCD.

3 Communication in Large-Scale LQCD Simulations

We have introduced the background of LQCD simulations in Section 2, now we delve into the details of the communication mechanisms in LQCD simulations. Figure 2 shows the communication mechanism in two dimensional LQCD simulations. Note that with enough computing resources, we can map one spinor to each core. For instance, if there were 256 cores in the system and the grid size were 256, then each spinor could be mapped to a core. After each round of computation, each spinor exchanges a SU3 vector with each of its n neighbors. For example, in Figure 6 at each communication point, the spinor receives a SU3 vector from one of its neighbors. Each SU3 vector consists of three pairs, or six floating point values. Therefore, each communication event involves the sending/receiving of six floating values.

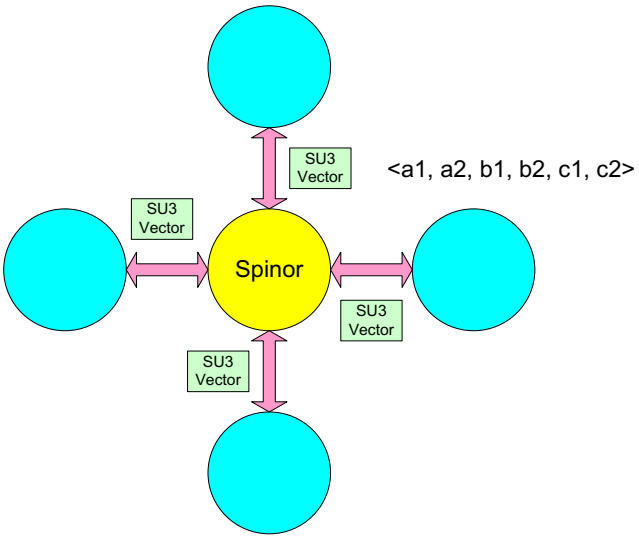

Fig. 2. Communication patterns of LQCD simulation

In LQCD simulations, communication happens frequently and incurs a very high overhead. To find out the communication frequency, we execute the LQCD simulation with a grid size of 2^8 on a four-core server, and we measured the time taken for each round of computation without communication. Our results indicate that on average a spinor takes only 1350 cycles to compute its results. During this time, the spinor requires one SU3 vector from each of its neighbors. In a two dimensional case, up to four communication events (one from each neighbor) may take place during one round of the spinor computation.

As for the communication overhead, Vranas *et al.* ported LQCD simulations onto the IBM BlueGene/L supercomputer and characterized its performance [5]. Their results indicate that communication is a major performance bottleneck of LQCD simulations. Specifically, when running a LQCD simulation with lattice size of 2^4, the performance reaches 31.5% of the peak performance with the communication turned off. On the other hand, with the communication turned on, the performance dramatically drops to 12.6% of the peak performance. In this case, communication contributes to more than 60% of performance degradation.

4 Value Prediction in LQCD Simulations

As shown in the previous section, communications happen frequently in LQCD simulations and incur a very high performance overhead. Particularly, these communications exhibit a special pattern, such that six, instead of one, floating point values are communicated in each communication event. This pattern complicates the design of value predictors, because now we have to correctly predict all six values instead of one. In this section we study the impacts of value prediction techniques on the performance of LQCD simulations

4.1 Value Predictability in LQCD Simulations

The feasibility of the value prediction techniques on a particular application depends on two factors: the predictability of data values, which determines the prediction accuracy; and the design of the value predictor, which determines the overheads taken to generate a prediction and to perform rollbacks when mispredictions occur.

The predictability of data values depends on the value redundancy existed in a particular application. The more redundant the values are, the easier to make a correct prediction. For example, given two data streams, one is {1,1,1,1,1,2}, the other is {2,5,1,7,0,9,3}, it is obvious that the first stream is easier to predict compared to the second one. In order to quantify the inherent data redundancy in the communication data of LQCD simulations, we follow the approach presented in [16] and use information theory measures to identify the prediction accuracy upperbound. Note that the prediction accuracy upperbound derived using this method is theoretical, and therefore not tied to any value predictor design.

As shown in Equation 1, information entropy quantifies the uncertainty involved in a random variable. Generally, the more redundant values exist in a data stream, then the less uncertain/random the data stream is, and it leads to a lower entropy number. In Equation1, $|S|$ represents the number of values in the data stream; $p(i)$ is the probability of the i^{th} value being produced; and log are base-2 logarithmic functions. The result $H(x)$ is the minimum number of bits required to represent all possible values in the selected data stream.

$$(1) \quad H(X) = -\sum_{i=1}^{|S|} p(i)\log(p(i))$$

Although information entropy quantifies the uncertainty of different data streams, they do not directly provide information about value prediction accuracy. To achieve this, Fano's inequality correlates information entropy and data value predictability by defining the lower bound on the prediction error probability, m, of any data stream. Then, from the lower bound of prediction error m, we can derive the upper bound for the prediction accuracy as $p = 1 - m$. Equation 2 defines the lower bound for the prediction error probability on independent value streams and Equation 3 defines $h(m)$ used in Equation 2.

$$(2) \quad m \geq \frac{H(X) - h(m)}{\log(|X|) - 1}$$

$$(3) \quad h(m) = -(m\log(m) + (1-m)\log(1-m))$$

To characterize the communication value predictability, we execute the LQCD simulation and randomly select four spinors and capture their communication values, then we apply the Equations 1, 2, and 3 on these values streams and the results are shown in Table 1: the predictability, or the maximum prediction accuracy, of individual values is over 50%; however, when we try to correctly predict all six values all together, the predictability dramatically drops to 27%. This observation is consistent across the selected spinors. This result may imply that value prediction has little benefit as the incorrect predictions (73% of all predictions) would result in rollbacks, which degrade the performance. However, in the next subsection, we show that by applying a

confidence estimator in our value predictor design, most of the mispredictions can actually be filtered out.

Table 1. Communication Value Predictability in LQCD Simulations

spinors	v1	v2	v3	v4	v5	v6	Total
8	55.30%	55.20%	55.10%	55.00%	55.30%	55.30%	27.50%
112	55.20%	55.20%	55.20%	55.20%	55.10%	55.10%	27.50%
199	57.00%	57.00%	57.00%	57.20%	57.20%	57.20%	29.00%
222	57.00%	57.00%	57.20%	57.20%	57.10%	57.10%	29.00%

4.2 Performance of Value Predictors on LQCD Simulations

Recall that the feasibility of the value prediction techniques depends on two factors: value predictability and the performance of value predictors. In the previous subsection we have identified the first factor, namely the predictability, or maximum prediction accuracy, that can be achieved on LQCD communication values. As for the second factor, to characterize the performance of an actual value predictor, we implement a Last Value Predictor (LVP) in the LQCD simulation program, which predicts the current value the same as the previous one. On top of the LVP, we also implement a confidence estimator, which is a 2-bit saturation counter that decides whether it is beneficial to use the predicted value for speculative execution. As shown in Figure 3, the confidence estimator operates as follows: with a correct prediction, the counter is incremented by one until it saturates; with an incorrect prediction, the counter is reset to zero; when the count is greater than or equal to two, then the predicted value is used; otherwise, no speculative execution is initiated. The purpose of this confidence estimator is to filter out those predictions that are not likely to succeed and to use those predictions that have a high probability of success. Same as in the previous experiment, we apply this design on four randomly selected spinors. In this case, at each communication point, the LVP generates a prediction, and then the confidence estimator decides whether to use the predicted value for speculative execution.

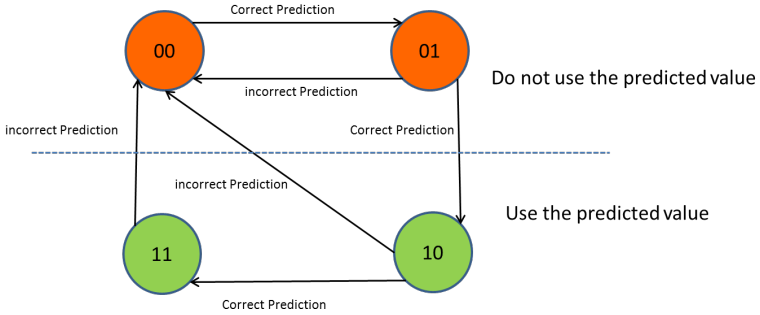

Fig. 3. Confidence estimator for value prediction

The results are shown in Table 2: the column *accuracy* shows the prediction accuracy achieved by the LVP. The column (*pred, conf*) shows the situation that the prediction is correct and the confidence estimator estimates the prediction is likely to succeed. In this case, the value prediction operation generates performance gain. On the other hand, the column (*!pred, conf*) shows the situation that the prediction is incorrect and the confidence estimator estimates the prediction is likely to succeed. In this case, an incorrectly predicted value would be used for speculative execution, thus resulting in a rollback. For the other two columns (*pred, !conf*) and (*!pred, !conf*), no predicted value is used and thus neither performance gain nor rollback would occur.

Table 2. Performance of the Last Value Predictor on LQCD simulations

Spinors	Accuracy	pred,conf	pred, !conf	!pred, conf	!pred, !conf
8	24.56%	21.40%	3.16%	3.16%	72.28%
112	24.70%	21.52%	3.18%	3.18%	72.12%
199	24.95%	22.82%	2.13%	2.13%	72.92%
222	25.00%	22.83%	2.14%	2.14%	72.89%
AVG	24.80%	22.15%	2.65%	2.65%	72.55%

The results show that on average the LVP design is able to achieve 24.8% of prediction accuracy. Specifically, 22.15% of the predictions result in performance gain; 2.65% of the predictions are correct but discarded by the confidence estimator; 2.65% of the predictions are incorrect but missed by the confidence estimator, thus resulting in rollbacks; and 72.55% of the predictions are incorrect and successfully filtered by the confidence estimator. In other words, 22.15% of the predictions result in performance improvement, and only 2.65% of the predictions result in performance degradation.

Note that we also implement the software stride value predictor (SVP), which predicts that the difference between the current value and the first previous value is the same as the difference between the first previous value and the second previous value; and the software finite context predictor (FCP), which maintains multiple contexts of past values, and when the current context matches one in the context table, we predict the next value the same as the next value maintained in that context. However, these predictors either similar performance compared to the LVP, thus we only present the results of the LVP.

4.3 Performance Impact of Value Prediction on Communication

In order to understand the performance impact of value prediction on communication, we construct an analytical model in Equation 4 to describe the communication latency with value prediction enabled: *com_lat* represents the communication latency, *mispred%* represents the percentage of mispredictions, *rollback%* represents the percentage of rollbacks, *pred_oh* represents the overhead incurred by each prediction operation, and *rb_oh* represents the overhead incurred by each rollback operation. Note that. Equation 4 identifies the break-even point where value prediction brings no performance gain nor performance degradation.

(4) $com\_lat = mispred\% \times com\_lat + pred\_oh + rollback\% \times rb\_oh$

To measure the value prediction and rollback overheads, we execute the software value predictor on our four-core server. In detail, each time the LVP is triggered, it generates six floating point values and consults the confidence estimator to decide whether the predicted value should be used, this process takes 40 clock cycles to complete. When the actual values are produced, they are compared against the predicted values. If they don't match, the program reloads the program context, and branches to the checkpoint. Each rollback process takes 50 cycles to complete. By plugging in the parameters obtained from LQCD simulations, we derive that as long as the communication latency is higher than 186.5 cycles, then value prediction would bring performance gain.

5 Reducing Overheads with Hardware Value Predictors

Since communications take place frequently in LQCD simulations, thus the 40-cycle software prediction overhead may significantly affect the overall performance. In this subsection we explore the hardware implementation of value predictor designs to accelerate the prediction process. Also, we identify the resource utilization and power consumption overheads of the hardware implementation.

To achieve this, we implement several value predictor designs onto the ML401 FPGA board [20]. These designs include the LVP, the SVP, the FCP, and the hybrid predictor (Hybrid), which is the combination of the three as well as a runtime decision logic to select the predictor to use in the next prediction. In order to measure the chip power consumption, we place a 0.033 Ohm shunt resistor on the power supply rail and measure the voltage drop across the shunt resistor. Then from the voltage drop we derive the current that goes into the Virtex-4 FPGA chip, and then we multiply this current by the 1.2 V supply voltage to calculate the chip power consumption under different configurations.

As summarized in Table 3, we implement these predictors on the FPGA board and measure the usage of hardware resources including the number of hardware slices (*#slices*), the number flip-flops (*#FF*), and the number of look-up tables (*#LUT*), also we show the performance (*Perf*) and the chip power consumption (*Power*) when the predictors are active. As a comparison, in the last row of Table 3, we also present the resource utilization and power consumption of a simple MIPS in-order processor [21].

Table 3. Hardware implementation of value predictors

	# slices	# FF	# LUTs	Perf (cycle)	Power (mW)
LVP	20	40	40	2	3.6
SVP	50	90	70	3	7.3
FCP	120	190	70	4	14.6
Hybrid	250	400	220	6	32.7
Processor	10450	10400	19500	N/A	650

The results show that the LVP uses only 20 slices, 40 flip-flops, and 40 look-up

tables, which respectively correspond to 0.2%, 0.4%, and 0.2% of the slice, FF, and LUT utilization of the simple MIPS in-order processor. Also, the LVP consumes 3.64 mW of power, which corresponds to 0.6% of the processor power consumption. As the more complicated predictors are used, the resource utilization and power consumption increases.

Our hardware LVP takes only 2 cycles to generate a prediction, which represents a 20-fold performance improvement compared to the software LVP design. In the first cycle, the hardware LVP generates a predicted value by accessing the history register, which stores the last value; also, it checks the confidence estimator. In the second cycle, based on the result of the confidence estimator, it either returns the value stored in the history register, or it sets the *No_Predict* signal to forfeit the operation. The design of SVP is slightly more complicated such that it takes 3 cycles to complete: in the first cycle, it checks the history register, which stores the last value; and it checks the delta register, which stores the difference between the previous two values. In the second cycle, it generates a prediction by adding the last value and the delta; also it consults the confidence estimator. In the third cycle, based on the result of the confidence estimator, it either returns the value stored in the history register, or it sets the *No_Predict* signal to forfeit the operation. The design of FCP is the most complicated one and it takes 4 cycles to complete: in the first cycle, it generates the current context by taking into account the last three values. In the second and third cycles, it accesses the context table with the current context in order to generate the predicted value; also it consults the confidence estimator. In the fourth cycle, based on the result of the confidence estimator, it either returns the value stored in the history register, or it sets the *No_Predict* signal to forfeit the operation. The *Hybrid* predictor is a combination of the three predictors with a decision circuit to decide which predictor to use. The hardware hybrid predictor takes 6 cycles to generate a prediction.

In our previous experiments, we have used LVP to generate predictions. By using the 2-cycle LVP latency as the *pred_oh* value in Equation 4, we derive that the value prediction technique would bring performance gain as long as the communication latency is higher than 13.5 cycles. Recall that the communication latency can be hundreds of cycles in many-core systems and even thousands of cycles in cluster environment, thus this result indicates that value prediction is an effective technique in reducing the communication overhead.

6 Improving Prediction Accuracy with Tolerance Predictor

Floating point numbers are used extensively in scientific simulations to approximate the actual values, and the accuracy of the floating point approximation is critical to the stability of the computation scheme. However, the full range of accuracy may not be always needed. For instance, the 32-bit floating point number format can represent values as small as 10^{-38}, and if the application would only require accuracy up to 10^{-20}, then the numeric scheme would be able to tolerate value mismatches that are less than 10^{-20}. Based on this condition, we design a tolerance predictor which tolerates a small discrepancy between the predicted value and the actual value, such that if the discrepancy is smaller than the tolerance range, then we still treat it as a correct

prediction. We apply this design on the LQCD communication values and the results are organized in Figures 4 and 5.

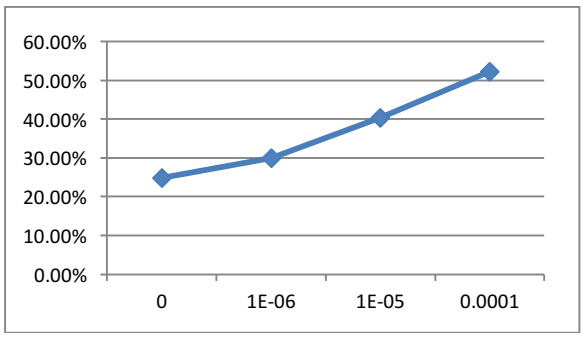

Fig. 4. Prediction accuracy of the tolerance predictor: the x-axis shows the tolerance

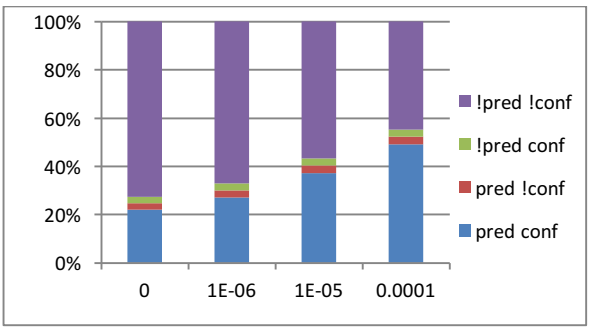

Fig. 5. Performance of the tolerance predictor: the x-axis shows the tolerance

Figure 4 indicates that as the tolerance range increases, the prediction accuracy also increases; when the tolerance range increases from 0 to 10^{-4}, the prediction accuracy also increases from 26% to 53%. In addition, Figure 5 shows that as the tolerance range increases, the percentage of predictions that lead to rollbacks remains constant, but the percentage of predictions that lead to performance gain increases. When the tolerance range is 10^{-4}, 48% of the predictions lead to performance gain and only 2.6% of the predictions lead to rollbacks, making value prediction a very attractive technique in reducing the communication latency.

For the tolerance predictor to be beneficial, it is essential to design the proper tolerance range such that it does not affect the stability of the numerical scheme. For example, as shown in Equation 5, the FTCS (Forward-Time Central-Space) method is a finite difference method used for numerically solving parabolic partial differential equations. Assume that this numerical scheme requires accuracy up to 10^{-20}, and that all values in this computation are represented using single-precision floating point numbers. In the first case, if the value to be predicted were U_j^n, then the tolerance range could be simply set to 10^{-20}, this is because the value of U_j^n directly affects the result U_j^{n+1}. On the other hand, if the value to be predicted were U_{j-1}^n, then the tolerance range

could be set to $10^{-20} \times 2 \times \Delta t / \Delta x$, such that the tolerance range depends on the resolution of the numerical scheme.

$$(5) \quad U_j^{n+1} = U_j^n + \frac{\Delta t (U_{j-1}^n - U_{j+1}^n)}{2\Delta x}$$

7 Conclusions and Future Work

In this paper we use LQCD simulations as a case study to understand how value prediction techniques can address the communication problems. As a first step, we implement a software value predictor. With this predictor, 22.15% of the predictions would result in performance gain, and only 2.65% of the predictions would result in rollbacks, the other 75.2% of the predictions are filtered out by the confidence estimator and do not affect performance. The software predictor design incurs a 40-cycle overhead to generate a prediction and each rollback takes 50 cycles. Since in LQCD simulations, communications happen frequently, this software prediction and rollback overhead may result in performance degradation. To address this problem, we design and implement the hardware value predictor and evaluate them on a FPGA board. The results show that the hardware predictors incur less than 0.5% of hardware overhead while reducing the prediction latency by up to 20 folds. In addition, for floating numbers, one very interesting property we observe is that as we relax the accuracy requirement, we could improve the prediction accuracy significantly. Based on this, we propose and implement an initial prototype of the tolerance value predictor. As the tolerance range increases, the prediction accuracy also increases dramatically.

The current paper naturally leads to two directions our future work: first, we have evaluated the value prediction techniques on a four-core server and the results are encouraging. Thus, we will extend the current work onto a large-scale computing platform and study how these techniques improve the performance of LQCD simulations in a cluster environment. Second, we feel that the tolerance predictor has a great potential to achieve high prediction accuracy. Therefore, we will study how the tolerance range can be set for different numerical schemes and how the tolerance predictor improves the performance of these numerical schemes.

References

1. K. Asanovi, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, K.A. Yelick "The Landscape of Parallel Computing Research: A View from Berkeley," EECS UC Berkeley, Technical Report No. UCB/EECS-2006-183, 2006.
2. J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, L. Peh "Research Challenges for On-Chip Interconnection Networks," IEEE Micro 27, 5, 96-108, 2007.
3. R.P. Martin, A.M. Vahdat, D.E. Culler, and T.E. Anderson "Effects of communication latency, overhead, and bandwidth in a cluster architecture," SIGARCH Comput. Archit. News 25, 2, 85-97, 1997.
4. J.G. Steffan, C.B. Colohan, A. Zhai, and T.C. Mowry "Improving Value Communication for Thread-Level Speculation," Proc. 8th international Symposium on High-Performance Computer Architecture, 2002.
5. P. Vranas, G. Bhanot, M. Blumrich, D. Chen, A. Gara, P. Heidelberger, V. Salapura, and J.C. Sexton "The BlueGene/L supercomputer and quantum ChromoDynamics." In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing

6. M.H. Lipasti and J.P. Shen "Exceeding the dataflow limit via value prediction," Proc. 29th International Symposium on Microarchitecture, 1996.
7. Y. Sazeides and J.E. Smith "The predictability of data values," Proc. 30th Annual International Symposium on Microarchitecture, 1997.
8. Y. Sazeides, S. Vassiliadis, and J.E. Smith "The performance potential of data dependence speculation and collapsing," Proc. 29th Annual International Symposium on Microarchitecture, 1996.
9. A. Sodani and G.S. Sohi "Understanding the differences between value prediction and instruction reuse," Proc. 31st Annual International Symposium on Microarchitecture, 1998.
10. D.M. Tullsen and J.S. Seng "Storageless value prediction using prior register values," Proc. 26th International Symposium on Computer Architecture, 1999.
11. A. Mendelson and F. Gabbay "Speculative execution based on value prediction," EE Department Technical Report 1080, Technion – Israel Institute of Technology, 1996.
12. P. Marcuello and A. Gonzalez "Thread partitioning and value prediction for exploiting speculative thread-level parallelism," IEEE Transactions on Computers, Vol. 53, No. 2, 2007.
13. J. Oplinger, D. Heine, and M.S. Lam "In search of speculative thread-level parallelism," Proc. 8th International Conference on Parallel Architectures and Compilation Techniques, 1999.
14. The Lattice Web: http://www.lqcd.org/
15. J. Parcerisa and A. González "Reducing wire delay penalty through value prediction," Proc. 33rd Annual ACM/IEEE international Symposium on Microarchitecture, 2000.
16. S. Liu, J-L. Gaudiot, Potential impact of value prediction on communication in many-core architectures, IEEE Transactions on Computers, Vol. 58, 6, June 2009
17. J.T. Feo "An analysis of the computational and parallel complexity of the Livermore loops," Parallel Computing, 7:163-185, 1988.
18. J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC Simulator," http://sesc.sourceforge.net, 2006
19. W. Zhu, V.C. Sreedhar, Z. Hu, and G. Gao "Synchronization State Buffer: Supporting Efficient Fine-grained Synchronization on Many-Core Architectures," Proc. 34th International Symposium on Computer Architecture, 2007.
20. Xilinx ML401 Overview, http://www.xilinx.com/products/boards/ml401/index.htm
21. The eMIPS Project: http://research.microsoft.com/en-us/projects/emips/default.aspx
22. S. Liu, C. Eisenbeis, and J-L. Gaudiot, "A Theoretical Framework for Value Prediction in Parallel Systems", Proc. of 39th International Conference on Parallel Processing, 2010.
23. R. M. Gray, *Entropy and Information Theory*, New York: Springer-Verlag, 1990.
24. F. Belletti, S.F. Schifano, R. Tripiccione, F. Bodin, P. Boucaud, J. Micheli, O. Pene, N. Cabibbo, S. de Luca, A. Lonardo, D. Rossetti, P. Vicini, M. Lukyanov, L. Morin, N. Paschedag, H. Simma, V. Morenas, D. Pleiter, F. Rapuano, "Computing for LQCD: apeNEXT", Computing in Science & Engineering, Volume 8, 2006.
25. L. Piccoli, J.B. Kowalkowski, J.N. Simone, X-H. Sun, H. Jin, J.D. Holmgren, N. Seenu, A.G. Singh. "Lattice QCD Workflows: A Case Study" Proc. 4th IEEE International Conference on eScience, 2008.
26. P. Vranas, M.A. Blumrich, D. Chen, A. Gara, M.E. Giampapa, P. Heidelberger, V. Salapura, J.C. Sexton, R. Soltz, G. Bhanot, "Massively Parallel Quantum ChromoDynamics" IBM Journal of Research and Development, Volume 52, 2008.
27. R. Ammendola, M. Guagnelli, G. Mazza, F. Palombi, R. Petronzio, D. Rossetti, A. Salamon, P. Vicini, "APENet: a high speed, low latency 3D interconnect network" Proc. IEEE International Conference on Cluster Computing, 2004.
28. IBM Blue Gene http://domino.research.ibm.com/comm/research_projects.nsf/pages/bluegene.index.html