Fault-Tolerant and Reliable Computation in Cloud Computing

Jing Deng[†] Scott C.-H. Huang[‡] Yunghsiang S. Han[*] and Julia H. Deng[§]
[†]Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC 27412, USA.
[‡]Department of Electrical Engineering, National Tsing Hua University, Hsinchu, 300 Taiwan.
[*]Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, 106 Taiwan.
[§]Intelligent Automation, Inc., Rockville, MD, USA.

***Abstract*— Cloud computing, with its great potentials in low cost and on-demand services, is a promising computing platform for both commercial and non-commercial computation clients. In this work, we investigate the security perspective of scientific computation in cloud computing. We investigate a cloud selection strategy to decompose the matrix multiplication problem into several tasks which will be submitted to different clouds. In particular, we propose techniques to improve the fault-tolerance and reliability of a rather general scientific computation: matrix multiplication. Through our techniques, we demonstrate that fault-tolerance and reliability against faulty and even malicious clouds in cloud computing can be achieved.**

I. Introduction

Cloud computing is heavily based on a more traditional technology: grid computing, which has been researched for more than 20 years. Cloud computing focuses on the sharing of information and computation in a large network of nodes, which are quite likely to be owned by different vendors/companies. It is believed that cloud computing has been one of the sources for success in several major companies such as Google and Amazon.

Cloud computing is expected to be the platform for next generation computing, in which users carry thin clients such as smart phones while storing most of their data in the cloud and submitting computing tasks to the cloud. A web browser serves as the interface between clients and the cloud. Operating system in web browsers allows the users to manage their data and computation tasks.

One of the main drivers for the interest in cloud computing is cost and reliability. As personal computers and their OS and software are becoming more and more complex, the installation, configuration, update, and removal of such computer systems require a significant amount of intervention time from the users or system managers. Instead, outsourcing the computation tasks eliminates most of such concerns. The cloud of computer grids provides such services on fee-based, which can be more cost-efficient for the users than purchasing, maintaining, and upgrading powerful servers. Furthermore, resource sharing improves overall resource usage.

Another reason is the adaptation of large number of mobile devices including smart phones, PDAs, and Netbooks. Such computing devices usually have high mobility and may only be powerful enough to run just a web browser. Currently, the users of these devices are not allowed to perform the full list of tasks that they normally do on regular computing devices such as desktop computers and servers. Cloud computing can provide a seamless interface to allow the users to point their web browsers to an address and perform the full list of usual computing tasks such as document preparation, data query, image processing, and scientific computation.

Overall, cloud computing brings the following three new aspects in computing resource management: infinite computing resources available on demand for the perspective of the end users; zero up-front commitment from the cloud users; and short-term usage of any high-end computing resources [1], [2].

When a client submits computation tasks toward the cloud, security of such computation naturally becomes a great concern. This is mainly because the data leave the client and the computation results will be returning from the cloud, which are usually considered out of the client's control. Computation security has many perspectives [3]. In this work, we focus on fault tolerance and reliability of scientific computation, in particular, matrix multiplication. Matrix multiplication serves as the foundation for many complex problem solving and optimization. An interesting point of privacy concern arises when the user does not want to send all computation tasks toward a single cloud.

We investigate the question of ensuring the correctness of matrix multiplication in cloud computing with the existence of faulty or even malicious clouds. In particular, we focus on the problem of dividing a complex matrix multiplication task into several subtasks and submitting them to different clouds for computation. Because different clouds have different cost and reliability, the selection of such clouds becomes an interesting problem for cost and reliability concerns.

This paper is organized as follows: in Section II, we summarize important related works. In Sections III, IV, and V, we investigate the issues of cloud selection and protection against faulty and malicious clouds. Section VI, we conclude the work.

II. Related Work

The landmark paper on fault-tolerant matrix operation was published in 1984 by Huang and Abraham [4], in which an Algorithm-Based Fault Tolerance (ABFT) method was proposed. ABFT uses matrix or vector level checksum in row and column to detect a faulty processor in multiple processor

systems. The method can be used to detect and correct errors in matrix operations such as addition, multiplication, scalar product, and LU-decomposition performed in multiple processor systems which may have one failed processor. In our work, however, we focus on the problem of achieving a certain reliability with the minimum cost in potentially faulty clouds.

In [5], Mei et al. investigated the problems of dynamic computing service registration, large data storage and access, adaptability, and quality discovery in cloud computing. A survey of trust and privacy in cloud computing was presented by Cachin et al. [3]. The survey focused on data storage in cloud grids and how to provide confidentiality, integrity, availability, and service guarantee. For instance, it is undesirable to test whether the cloud grids have really stored all of a large amount of data by retrieving the entire data. Instead, a subtle and much more efficient proof can used to achieve the goal of "proof of retrievability" or "proof of data possession".

Cloud computing is closely related to grid computing. Chakrabarti et al. discussed security issues in grid computing in [6]. Since clouds may be formed by multiple grids from different entities, it then becomes critical to address the security issues such as confidentiality, integrity, and service availability. In [7], privacy-friendly client cloud computing were presented. Wang et al. proposed to use erasure correction coding in file preparation toward cloud computing storage [8].

III. Cloud Selection

We assume that there are multiple clouds and each cloud contains multiple servers. The servers on these clouds are only trusted partially depending on experiences of the client him/herself and other. We further assume that each of the cloud's reliability (either through experience or through recommendations) and cost are known to the client.

Though there exist many applications for cloud computing, we focus on scientific computation, in particular, large matrix multiplication. It can be shown that many complex scientific computation can be generalized to large matrix multiplications.

Let $\boldsymbol{a}_i$ denote i-th **row** of A and $\boldsymbol{b}_j^T$ j-th **column** of B. Then $D = A \times B$ with dimension $\ell \times n$ is the target matrix which is to be computed by clouds (see Fig. 1). Matrix A has dimension of $\ell \times m$ and matrix B has dimension of $m \times n$. Assume that a client wants to perform a large matrix multiplication, $D = A \times B$, over L clouds. The clouds might have different costs and reliability values. While reliability is referred to as how reliable and secure a cloud is, cost can represent a number of objectives: network communication cost, computation cost, monetary charge for the use of a cloud's service.

The goal of the client is to minimize its cost under pre-specified reliability constraint. Without loss of generality, assume that $R_1 \leq R_2 \leq \cdots \leq R_L$, where R_i is the reliability of cloud i and $i = 1, 2, \cdots, L$. Let C_i be the cost of cloud i to perform one row of A multiplying B. Usually, a cloud with better reliability should have higher cost, but it is not always true. The values of reliability and cost of a cloud can be obtained through published price and/or past experience of other users with the cloud. Also, cloud providers may provide various options of clouds with different charge of usage and different reliability (e.g., faulty rate).

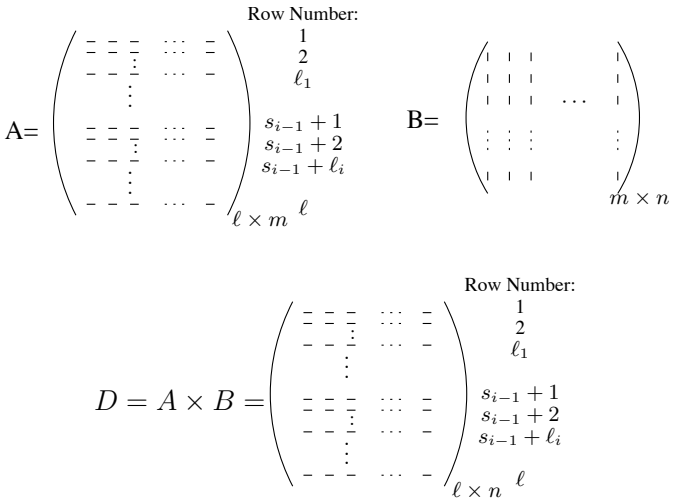

Fig. 1. The computation of $D = A \times B$ through the clouds.

Problem Definition: Assume that the client dispatches ℓ_i rows of A to be multiplied by B in cloud i. The overall cost for this task is then

$$C = \sum_{i=1}^{L} C_i \ell_i \tag{1}$$

and the reliability of the dispatched task is

$$R = (\sum_{i=1}^{L} R_i \ell_i)/\ell \ , \tag{2}$$

where ℓ is the number of rows of A and $\sum_{i=1}^{L} \ell_i = \ell$. Our objective is as follows: *minimize C, subject to $R \geq R_s$, where R_s is a pre-specified reliability requirement*.

Note that a different overall reliability model other than (2) may result in a much simpler solution. For example, if the overall reliability of the computation is defined as the lowest value of the reliability of the all clouds working on the computation, i.e.,

$$R \stackrel{\text{def}}{=} \min_{i=1,2,\cdots L} \{R_i\} \tag{3}$$

then a simple assignment is to exclude all clouds with reliability lower than R_s. For the rest of the clouds which have reliability value higher than R_s, we can choose those with lowest cost C_i first and use those clouds with higher costs only if necessary. It is trivial to prove that this choice is optimum.

It is easy to see that searching an optimal solution for the above problem is to solve an integer programming problem. The problem can be rewritten to a standard integer program-

TABLE I

EXAMPLE OF DISPATCHING JOBS TO THREE CLOUDS WITH DIFFERENT RELIABILITY AND COST. THE JOB HAS $\ell = 10$ ROWS OF MULTIPLICATION AND THE RELIABILITY REQUIREMENT $R_s = 2.6$. IT TURNS OUT THAT THE BEST ASSIGNMENT IS $(2, 8, 0)$ AS SHOWN IN ASSIGNMENT 6. THESE NUMBERS ARE FOR ILLUSTRATION PURPOSE ONLY. REAL RELIABILITY VALUES COULD BE DIFFERENT TO WHAT ARE SHOWN HERE.

(R, C)	Cloud1 (1,1)	Cloud2 (3,2)	Cloud3 (4,3)	Overall
Assignment 1	2	0	8	(3.4, 26)
Assignment 2	10	0	0	(1, 10)
Assignment 3	5	0	5	(2.5, 20)
Assignment 4	0	10	0	(3, 20)
Assignment 5	4	2	4	(2.6, 20)
Assignment 6	2	8	0	(2.6, 18)

ming problem as follows:

$$\begin{array}{ll} \text{Minimize} & C = \sum_{i=1}^{L} C_i \ell_i \\ \text{Subject to} & R = \left(\sum_{i=1}^{L} R_i \ell_i\right) \Big/ \ell \geq R_s \\ & \sum_{i=1}^{L} \ell_i = \ell \quad \ell_i \text{ is integer .} \end{array} \tag{4}$$

Any standard tool for solving integer programming can be used to solve (4), e.g., lp_solve [9].

We present an example of three clouds in Table I, which shows several different assignments. Next we present some properties of our integer programming problem which can help to fasten the search procedure.

We have the following lemma regarding to those clouds with higher cost but lower reliability:

Lemma 1: If there exist two clouds i and j such that $R_j \geq R_i$ and $C_j < C_i$, then cloud i should be ignored in the assignment process.

Proof: Without loss of generality, we assume $i < j$. Assume that there is an optimal assignment $\mathcal{A} = \{\ell_1, \cdots, \ell_i, \cdots, \ell_j, \cdots, \ell_L\}$ such that $\ell_i > 0$. Construct a new assignment $\mathcal{A}' = \{\ell'_1, \cdots, \ell'_i, \cdots, \ell'_j, \cdots, \ell'_L\}$, where all elements are the same with $\mathcal{A}$ except ℓ_i and ℓ_j:

$$\begin{aligned} \ell'_j &= \ell_i + \ell_j \\ \ell'_i &= 0 \ . \end{aligned} \tag{5}$$

Therefore, the cost and reliability of assignment $\mathcal{A}'$ can be expressed as

$$C(\mathcal{A}') = \sum_{k=1}^{L} C_k \ell'_k = \sum_{k=1}^{L} C_k \ell_k + C_j \ell_i - C_i \ell_i < C(\mathcal{A}) \tag{6}$$

and

$$\begin{aligned} R(\mathcal{A}') &= \sum_{k=1}^{L} R_k \ell'_k / \ell \\ &= \sum_{k=1}^{L} C_k \ell_k / \ell + R_j \ell_i / \ell - R_i \ell_i / \ell \\ &= R(\mathcal{A}) + (R_j - R_i)\ell_i / \ell \\ &\geq R(\mathcal{A}) \ , \end{aligned} \tag{7}$$

where we have used the inequalities of $C_j < C_i$ and $R_j \geq R_i$.

Since assignment $\mathcal{A}'$ has a lower cost and a reliability that is at least as good as $\mathcal{A}$, it contradicts to the assumption that $\mathcal{A}$ is an optimal assignment. Hence, it is obvious that cloud i should be ignored.

■

Corollary 1: If there exist two clouds i and j such that $R_j > R_i$ and $C_j \leq C_i$, then cloud i can be ignored in the assignment process.

The proof of Corollary 1 is similar to that of Lemma 1 and therefore omitted. Please note that the difference between Lemma 1 and Corollary 1 is the location of the equal sign. Therefore, we can assume a strictly increasing function in C_i and R_i, where $i \in \{1, 2, \cdots, L\}$. That is, $C_1 < C_2 < \cdots < C_L$ and $R_1 < R_2 < \cdots < R_L$.

It is obvious that there is no solution if $R_s > R_L$. Hence, in this section we assume that $R_s \leq R_L$. Denote the optimal assignment $\mathcal{A}^* = \{\ell_1^*, \cdots, \ell_L^*\}$. If $R_s \leq R_1$, then $\ell_1^* = \ell$ and $\ell_i^* = 0$ for $i \neq 1$. We have the following lemmas regarding to the case $R_1 < R_s \leq R_L$.

Lemma 2: Given $R_s \geq R_t$, $t \in \{1, 2, \cdots, L-1\}$. Then

$$\sum_{i=1}^{t} \ell_i^* \leq \left\lfloor \frac{R_L - R_s}{R_L - R_t} \ell \right\rfloor \ , \tag{8}$$

where $\lfloor x \rfloor$ is the largest integer that is no more than x.

Proof: The reliability of the assignment $\mathcal{A}^*$ can be computed as

$$\begin{aligned} R &= \sum_{i=1}^{L} \ell_i^* R_i / \ell \\ &= \sum_{i=1}^{t} \ell_i^* R_i / \ell + \sum_{i=t+1}^{L} \ell_i^* R_i / \ell \\ &\leq \sum_{i=1}^{t} \ell_i^* R_t / \ell + \sum_{i=t+1}^{L} \ell_i^* R_L / \ell \\ &= \left(\sum_{i=1}^{t} \ell_i^*\right) \cdot R_t / \ell + \left[\ell - \left(\sum_{i=1}^{t} \ell_i^*\right)\right] \cdot R_L / \ell \ . \end{aligned} \tag{9}$$

Using inequality $R \geq R_s$ and re-arranging (9), we have

$$\sum_{i=1}^{t} \ell_i^* \leq \frac{R_L - R_s}{R_L - R_t} \ell \ . \tag{10}$$

■

And we have the following lemma:

Lemma 3: Given $R_s \leq R_{t+1}$, $t \in \{1, 2, \cdots, L-1\}$. Then

$$\sum_{i=1}^{t} \ell_i^* \geq \left\lceil \frac{R_{t+1} - R_s}{R_{t+1} - R_1} \ell \right\rceil \ , \tag{11}$$

where $\lceil x \rceil$ is the smallest integer that is no less than x.

Proof: Detailed proof is omitted due to page limit, but outlined below: an optimum assignment is assumed first. Then a new assignment will be constructed based on the optimum assignment but with a lower cost. ■

In the example discussed in Table I, we can see from Lemma 2 that $\ell_1^* < \frac{4-2.6}{4-1} \times 10 = 4.3$. Therefore, the maximum allowable value for ℓ_1^* is 4. Based on Lemma 3, we can see that $\ell_1^* \geq \frac{3-2.6}{3-1} \times 10 = 2$. The minimum value for ℓ_1^* is 2. Note that $t = 1$ in this example, with $R_1 < R_s < R_2$.

When $R_L < R_s$, there is no solution for the above integer programming problem. In this case, we need to add redundancy to increase reliability of each cloud. Since the client has no control over which grids to choose from, it needs to add redundancy by itself to increase reliability of computation results. Next we present a method to increase reliability of computation results by error-detection codes.

IV. Protection Against Faulty Clouds

In this section, we propose a countermeasure against faulty clouds. We will show that, by appropriately choosing certain system parameters, the detection failure probability of our scheme (the probability that a cloud returns incorrect results and the results are undetected by the client) is negligible. Without loss of generality, we assume that the $(s_{i-1}+1)$-th, $(s_{i-1}+2)$-th,..., s_i-th rows of A are dispatched to cloud i, where $s_i = \sum_{j=1}^{i} \ell_j$ with $s_0 = 0$. Let $\boldsymbol{r}_i = (r_{i1}, r_{i2}, \ldots, r_{i\ell_i})$, $r_{ij} \in \{1, -1\}$ for $j = 1, 2, \ldots, \ell_i$, be randomly generated sequences with length ℓ_i, where $i = 1, \ldots, L$. The client first multiplies from the $(s_{i-1}+1)$-th row to s_i-th row of A by $\boldsymbol{r}_i$ to generate a parity-check row, $\boldsymbol{c}_i$, for Cloud i. That is,

$$\boldsymbol{c}_i = \sum_{j=1}^{\ell_i} r_{ij} \boldsymbol{a}_{s_{i-1}+j}, \ i = 1, \ldots, L \ . \tag{12}$$

Note that $\boldsymbol{c}_i$ is a row array with m elements.

Since all elements of $\boldsymbol{r}_i$ are 1 or -1, the computation involved in (12) is only addition and subtraction. Usually L is also much smaller than ℓ such that $\boldsymbol{c}_i B$, $i = 1, 2, \cdots, L$ can be performed in the most reliable cloud or even at client itself. Let

$$\boldsymbol{y}_i = \boldsymbol{c}_i B \ . \tag{13}$$

By (12), $\boldsymbol{y}_i$ can also be expressed as

$$\boldsymbol{y}'_i \ = \ \sum_{j=1}^{\ell_i} r_{ij} \boldsymbol{d}_{s_{i-1}+j} \ , \tag{14}$$

where $\boldsymbol{d}_j$ is the j-th row of matrix D.

The client proceeds in the following way: it first computes $\boldsymbol{y}_i$ using (12) and (13). When the rows of D are received from Cloud i, the client computes $\boldsymbol{y}'_i$ based on (14). If $\boldsymbol{y}_i$ and $\boldsymbol{y}'_i$ match with each other, the results from Cloud i are determined to be error-free. Otherwise, errors must have occurred. The job originally sent to the i-th cloud should be sent to the cheapest non-faulty cloud with reliability no less than i-th cloud. After the client obtains new results from the cloud, it repeats the above fault-tolerant procedure. This process lasts until all results passed the check or no non-faulty cloud can be used to recompute new results.

The fault-tolerance strength is dependent on the probability that (13) and (14) are equal given that at least one $\boldsymbol{a}_{s_{i-1}+j} B \neq \boldsymbol{d}_{s_{i-1}+j}$. We compute the detection failure probability as follows: in each column of $\boldsymbol{y}_i$, no matter how many faulty computational results generated by Cloud i, the only way to generate a correct column in $\boldsymbol{y}_i$ is that the last element matches the difference between $\boldsymbol{y}_i$ and sum of rest of computational results. Hence, the probability that the client cannot detect the faulty results generated by any cloud in this column is $\frac{1}{2^W}$, where W is the number of bits to represent any element in A (B). Note that there are n columns in $\boldsymbol{y}_i$ and all the elements in $\boldsymbol{y}_i$ and $\boldsymbol{y}'_i$ must match. Therefore, the detection failure probability, which defined as the probability that the i-th cloud returns incorrect results, but the client cannot detect the error, is

$$P_{MD} = \frac{1}{2^{nW}} \ . \tag{15}$$

Obviously, when W is reasonably large, the value of P_{MD} approaches 0.

V. Protection Against Malicious Clouds

While the above scheme works well against random faulty clouds, it does not protect the client from malicious clouds. This is mainly due to the overly simplified correctness check. For instance, a malicious cloud can intentionally increase all the elements in two rows by the same amount. If the corresponding r_{ij} of these two rows are different, the client cannot detect such intentional changes. The chance of any two r_{ij} being different is $1/2$, hence, the chance of a malicious cloud successfully fooling the client to accept incorrect results is $1/2$.

We can explain intuitively why modifying two rows works best for the malicious cloud. Suppose the malicious cloud guesses several coefficients r_{ij}. It will certainly be better to just guess two out of these coefficients and leave the rest alone. This is because of the larger coefficient space.

We have the following theorem and proof.

Theorem 1: The chance of a malicious cloud tricking the client to accept a result other than the correct one can be as high as $1/2$.

Proof: Assume that Cloud i is malicious and it modified[1]

$$\boldsymbol{a}_{s_{i-1}+m_1}, \boldsymbol{a}_{s_{i-1}+m_2}, \ldots, \boldsymbol{a}_{s_{i-1}+m_q}.$$

Let $\Delta_{s_{i-1}+m_1}, \Delta_{s_{i-1}+m_2}, \ldots, \Delta_{s_{i-1}+m_q}$ be the corresponding differences between modified rows and original rows. If $(r_{im_1}, \ldots, r_{im_q})$ is a solution to

$$\sum_{j=1}^{q} x_j \Delta_{s_{i-1}+m_j} = 0, \tag{16}$$

then Cloud i successfully forges the computational results which cannot be detected by the client. Next assume that $|\Delta_{s_{i-1}+j}| \neq |\Delta_{s_{i-1}+k}|$ for $k = 1, 2, \cdots, q$ and $k \neq j$. Let $(r_{im_1}, \ldots, r_j, \ldots, r_{im_q})$ be any solution of (16). Then $(r_{im_1}, \ldots, -r_j, \ldots, r_{im_q})$ is not a solution and any other

[1]Cloud i can modify B or even $\boldsymbol{d}_i$. However, these modifications are equivalent to a modification on a subset of rows of A sent to it.

solutions differ from it by at least two positions. Hence, any solution of (16) is one-to-one corresponding to a non-solution such that the number of solutions to (16) is no more than half of the total number of possible values for $(r_{im_1}, \ldots, r_{im_q})$. That is, the probability that Cloud i to successfully forge the computational results is no more than $1/2$. Next we consider the case $|\Delta_j| = |\Delta_k|$ for $j, k = 1, 2, \cdots, q$. It is easy to see that if $(r_{im_1}, \ldots, r_j, \ldots, r_{im_q})$ is a solution of (16), then $(r_{im_1}, \ldots, r_j, \ldots, -r_{im_q})$ is a solution of

$$\sum_{j=1}^{q-1} x_j \Delta_{s_{i-1}+m_j} + x_q(-\Delta_{s_{i-1}+m_q}) = 0 \ .$$

Hence, we only need to evaluate the number of solutions to

$$\sum_{j=1}^{q} x_j \Delta = 0 \ ,$$

where $\Delta = |\Delta_{s_{i-1}+m_j}|$ for $j = 1, 2, \cdots, q$ which is at most half of the total number of possible values for $(r_{im_1}, \ldots, r_{im_q})$. That is, a malicious cloud i can modify computation results and trick the client to accept it with a probability as high as $1/2$. ■

Note that this does not mean that the malicious cloud can always trick the client to accept any arbitrary result with probability of $1/2$. The result has to be maneuvered in a certain way in order to achieve such a surprisingly high success probability, e.g., modifying several rows in a certain way.

In order to protect the client's computation from such attacks from malicious clouds, we propose to extend the parity-check row given in (12) to k rows. Similarly to (12), we define a matrix R_i with dimension of $k \times \ell_i$.

$$C_i = R_i[\boldsymbol{a}_{s_{i-1}+1} \ \boldsymbol{a}_{s_{i-1}+2} \ \cdots \ \boldsymbol{a}_{s_{i-1}+\ell_i}]^T, \tag{17}$$

where $R_i = [r_{mj}]$, $m = 1, 2, \cdots k$, $j = 1, 2, \cdots, \ell_i$, and superscript T represents transposition operation. Note that the computation of $C_i B$ can be sent to another cloud for the sake of error isolation and malicious cloud detection. For example, these k rows of contents can be simply attached to the end of the ℓ_j row of A matrix for cloud j.

Since now there are k parity checks instead of 1, it is more difficult for a malicious cloud to tamper with the computation results and trick the client to accept them. Similarly, a malicious cloud's best chance is to change two rows in the results. Without loss of generality, we can assume that the malicious Cloud i modifies the first two rows of A sent to it with quantity Δ. By a similar argument to one parity check case,

$$r_{m1}\Delta + r_{m2}\Delta = 0, \text{ for } m = 1, 2, \ldots, k \tag{18}$$

is a necessary condition for the client to accept the tampered results. Since all r_{mj}, $m = 1, 2, \cdots, k$, $j = 1, 2, \cdots, \ell_i$, are randomly generated with values $1, -1$, the probability to satisfy (18) is $(1/2)^k$. In fact, we have the following theorem:

Theorem 2: Assume the k-parity check is implemented by the client. If a malicious cloud tries to tamper with the computation results, the chance that it succeeds is at most $(1/2)^k$.

Proof:

Assume that $|\Delta_{s_{i-1}+j}| \neq |\Delta_{s_{i-1}+k}|$ for $k = 1, 2, \cdots, q$ and $k \neq j$. By an argument similar to the case of one parity check, we know that the probability of successfully forging computational results is no more than

$$\frac{1}{1 + 2^{q(k-1)}}, \tag{19}$$

where q is the number of rows of A that are modified by the malicious cloud. Since $q, k \geq 2$,

$$\frac{1}{1 + 2^{q(k-1)}} \leq \frac{1}{1 + 2^{2(k-1)}} \leq \frac{1}{1 + 2^k} \ . \tag{20}$$

Therefore, (19) is no greater than $(1/2)^k$. Similarly, we can prove that the probability is no more than $(1/2)^k$ when $\Delta = |\Delta_{s_{i-1}+m_j}|$ for $j = 1, 2, \cdots, q$. We have just proven that the best chance of a malicious cloud tricking the client to accept a tampered result is $(1/2)^k$. ■

VI. Conclusions

Cloud computing is expected to the next generation computation platform for commercial and non-commercial users. In this work, we have investigated the fault tolerance and reliability issues of cloud computing in scientific computation, in particular matrix multiplication. Our analysis showed that, with careful design and cloud selection, computation in the cloud can be fault-tolerant and reliable.

In our future work, we will provide simulation performance of our scheme and compare it with other related schemes. We will also investigate the fault tolerance and reliability issues in a broader range of computations. The issue of privacy protection of the client's data and results will be studied as well.

References

[1] M. Armbrust, A. Fox, and et al., "Above the clouds: A berkeley view of cloud computing," UC Berkeley, Tech. Rep. UCB/EECS-2009-28, February 2009.

[2] K. Birman, G. Chockler, and R. van Renesse, "Toward a cloud computing research agenda," *SIGACT News*, vol. 40, no. 2, pp. 68–80, 2009.

[3] C. Cachin, I. Keidar, and A. Shraer, "Trusting the cloud," *ACM SIGACT News*, vol. 40, no. 2, pp. 81–86, June 2009.

[4] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computer*, vol. 33, no. 6, pp. 518–528, 1984.

[5] L. Mei, W. Chan, and T. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues," *Asia-Pacific Conference on Services Computing. 2006 IEEE*, vol. 0, pp. 464–469, 2008.

[6] A. Chakrabarti, A. Damodaran, and S. Sengupta, "Grid computing security: A taxonomy," *IEEE Security and Privacy*, vol. 6, no. 1, pp. 44–51, 2008.

[7] M. J. Miller and N. H. Vaidya, "Leveraging channel diversity for key establishment in wireless sensor networks," in *Proc. of the 25th Conference of the IEEE Communications Society (Infocom '06)*, Barcelona, Spain, April 23-29 2006.

[8] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," IIT, Tech. Rep., 2009.

[9] http://lpsolve.sourceforge.net/5.5/.