Prediction of Optimal Parallelism Level in Wide Area Data Transfers

Esma Yildirim, Dengpan Yin, Tevfik Kosar, *Member, IEEE*

Abstract—Wide area data transfers may be a major bottleneck for the end-to-end performance of distributed applications. A practical way of increasing the wide area throughput at the application layer is using multiple parallel streams. Although increased number of parallel streams may yield much better performance than using a single stream, overwhelming the network by opening too many streams may have an inverse effect. The congestion created by excess number of streams may cause a drop down in the throughput achieved. Hence, it is important to decide on the optimal number of streams without congesting the network. Predicting this 'optimum' number is not straightforward, since it depends on many parameters specific to each individual transfer. Generic models that try to predict this number either rely too much on historical information or fail to achieve accurate predictions. In this paper, we present a set of new models which aim to approximate the optimal number with least history information and lowest prediction overhead. An algorithm is introduced to select the best combination of historic information to do the prediction for evaluation purposes as well as optimizing prediction by reducing error rate. We measure the feasibility and accuracy of the proposed prediction models by comparing to actual GridFTP data transfers by using little historical information and have seen that we could predict the throughput of parallel streams accurately and find a very close approximation of the optimal stream number.

Index Terms—Data intensive distributed applications, wide area data transfers, parallel TCP streams, throughput prediction, performance modeling.

1 Introduction

The end-to-end performance of a data intensive distributed application heavily depends on the wide area data transfer performance and the effective throughput achieved by the application. Prediction of the effective throughput that is achievable by an application given the capacity of a network and current load is a study area in which several different methods have been developed either in high or low-level. As an example of low level methods, different transport protocols have been developed [1], [2], [3], [4] and also tuning the existing protocol parameters [5], [6], [7] gave promising results. Among those protocols, TCP is the most widely used one and different versions of TCP are implemented to increase efficiency in achievable transfer rate. On the high level, other techniques are proposed which use the existing underlying protocol. Opening parallel streams is one way of doing that and is widely used in many application areas from data-intensive scientific computing to live multimedia, and peer-to-peer paradigms.

It is shown that parallel streams achieve high throughput by mimicking the behavior of individual streams and get an unfair share of the available bandwidth [8], [5], [9], [10], [1], [2], [11]. On the other hand, using too many simultaneous connections results in congestion and the achievable throughput starts to drop down. Unfortunately it is difficult to predict the point of congestion and is variable over some parameters which are unique in both time and domain. The prediction of the optimal number of streams is hence very challenging and almost impossible without some parameters of current network conditions such as available bandwidth, RTT, packet loss rate, bottleneck link capacity and data size.

It is easier to optimize the network parameters in the case of bulk data transfers, where enough historical information is gathered and the parameters are optimized using this history information. This type of optimization has been already used with Stork data scheduler [6] before. However, for individual data transfers, the optimization becomes more challenging. Instead of relying on historical information, the transfer should be optimized based on instant feedback. In our case, this optimization is achieving optimal number of parallel streams to get the highest throughput. However, an optimization technique not relying on historical data in this case must not cause overhead of gathering instant data that is larger than the speed up gained with multiple streams for a particular data size.

The studies that try to find the optimal number of streams are so few and they are mostly based on approximate theoretical models [12], [13], [14], [3], [4]. They all have specific constraints and assumptions. Also the correctness of the proposed models are mostly proved with simulation results only. Hacker et al [12] claim that the total number of streams behaves like one giant stream that transfers in capacity of total of each streams' achievable throughput. However, this model only works for uncongested networks. Thus it does not give an answer to at what point the network will be congested. Crowcroft et al [3] declare the same theory but develop a protocol which at the same time provides fairness. Dinda et al [13] model the bandwidth of multiple streams as

• *E. Yildirim, D. Yin, and T. Kosar are with the Department of Computer Science and Center for Computation & Technology, Louisiana State University, Baton Rouge, LA, 70803.*

a partial second order equation and needs two different throughput measurement of different stream numbers to predict the others. In another model, Altman et al [14] claim that the total throughput always shows the same characteristics depending on the capacity of the connection as the number of streams increases and 3 streams are sufficient to get a 90% utilization. In [4], Kola et al present a new protocol that adjusts sending rate according to calculated backlog which provides a model to predict the current number of flows which could be useful to predict the future number of flows.

In this study, we present theoretical models that are used to predict the behavior of parallel streams and discuss their assumptions in application. We also apply those models and measure their accuracy against actual GridFTP transfers with the improvements we have made. It has been observed that GridFTP transfers show different characteristics in existence or absence of congestion due to opening too many streams and none of the existing models could predict this behavior. With the improvements we make on the existing models, with little historical information or instant prediction information we are able to predict this behavior and our tests have proven the correctness of our approach. With this information at hand any data scheduler may optimize its each individual transfer with little information to be gathered.

2 Existing Models

In this section, we present the existing models for predicting the behavior of parallel TCP streams and discuss their advantages, shortcomings and ease of applicability.

2.1 Hacker et al Model

In this model, the achievable throughput depends on three parameters: round trip time, packet loss rate and maximum segment size. The maximum segment size is in general IP maximum transmission unit (MTU) size - TCP header. Round trip time is the time it takes for the segment to reach the receiver and for a segment carrying the generated acknowledgment to return to the sender. The packet loss rate is the ratio of missing packets over total number of packets. The following formula represents an upper bound on the achievable throughput [12]:

$$Th <= \frac{MSS}{RTT}\frac{c}{\sqrt{p}} \tag{1}$$

Th represents the achievable throughput by an application opening single stream, MSS represents the maximum segment size, RTT is the round trip time, p is the packet loss rate and c is a constant. Of MSS, RTT, and p variables, packet loss is the most dynamic one while MSS is the most static one. According to TCP's AIMD congestion avoidance algorithm, packet loss is considered as an indication of congestion and the window size of TCP is halved immediately. Then it increases linearly until another packet loss event occurs. This gives a saw tooth behavior to the TCP congestion window [12].

An application opening n connections actually gains n times the throughput of a single connection, assuming all connections experiencing equal packet losses. Also the RTTs of all connections are equivalent since they most likely follow the same path. In that case, Equation 1 is rearranged for n streams as:

$$Th_n <= \frac{MSS \times c}{RTT}\left(\frac{n}{\sqrt{p}}\right) \tag{2}$$

However this equation accepts that packet loss is stable and does not increase as the number n increases. At the point the network gets congested, the packet loss rate starts to increase dramatically and the achievable throughput starts to decrease. So it is important to find that point of knee in packet loss rate. One possible way is to gather statistical data to apply a time series prediction of p, MSS and RTT. The requirement of that model is to gather data regarding RTT, MSS, and p. There are separate tools that give this information however, in that study Web100 is used to collect it [12].

2.2 Dinda et al Model

The model presented in the previous section is only valid for uncongested networks and does not give us a relation between packetloss rates and the number of streams. Considering both p_n and RTT_n all depend on the number of streams, if a model can be presented that computes this relationship with only small number of measurements, then throughput of n streams can be predicted. The model proposed by Dinda et al tries to solve Equation 2 by computing the relationship between p, RTT, and n. MSS and c are considered as constants. The relationship is represented by a new variable p'_n which is equalized to a partial second order polynomial which is believed to be best suited [13]:

$$p'_n = p_n\frac{RTT_n^2}{c^2 MSS^2} = a'n^2 + b' \tag{3}$$

After placing p'_n in Equation 2, total throughput of n streams is calculated as follows:

$$Th_n = \frac{n}{\sqrt{p}'_n} = \frac{n}{\sqrt{a'n^2 + b'}} \tag{4}$$

a' and b' are parameters to be found by measurements. To solve this equation, two achievable throughput measurements for two different parallelism levels are needed. The only possible way to find those throughput values is either use a tool that has the capability to do parallel transfers or to use information from past transfers.

$$Th_{n_1} = \frac{n_1}{\sqrt{a'n_1^2 + b'}} \tag{5}$$

$$Th_{n_2} = \frac{n_2}{\sqrt{a' n_2^2 + b'}} \quad (6)$$

By using equations 5 and 6, the values of a' and b' are calculated and placing them to Equation 4, the aggregate throughput for any number of streams can be calculated

n_1 and n_2 are two parallelism levels of which achievable throughput needs to be measured and n is the parallelism level of which achievable throughput will be predicted. Besides using partial second order equations, they also use linear and Full Second Order equations and based on experimental results partial second order equation is chosen. The experimental results claim that the parallelism level correctly can be predicted, however in those results the aggregated throughput of connections increases and then becomes stable but never falls down. So, it will be interesting to analyze the behavior of that function when the throughput starts to decrease.

By predicting the achievable throughput of n streams, it should be decided which stream number will be used. For this, a percentage is given as input for cross traffic effect. In that case, we may find the optimal number of streams that will effect the cross traffic by x%. For measurement of sample throughput values for two different parallelism levels that will be fed into the function, they use Iperf which has the capability to open parallel connections. The predicted results however are again compared to actual Iperf measurements and the behavior of actual protocol transfers are not considered.

2.3 Altman et al Model

The third model bases its correctness on the fact that opening too many connections imposes a processing overhead and leaves smaller bandwidth to other flows. Only 3 streams are enough to get a 90% link utilization. It is noted that for single stream case, a TCP connection may utilize only 75% of the bandwidth because of its AIMD (additive increase and multiplicative decrease) property. Only with parallel streams we could increase this ratio. Because only a small subset of the connections undergoes multiplicative decrease during congestion, this helps parallel streams to recover quickly. Assuming only one of the connections undergoes a multiplicative decrease at any time, the following formula can be used to predict the throughput [14]:

$$Th_n = C\left(1 - \frac{1}{1 + \frac{1+B}{1-B}n}\right) \quad (7)$$

B is $1/2$ for TCP connections as it decreases its window by half for the multiplicative decrease and C is the capacity of the link. There are two issues that need to be solved for that approach to be applied. First of all, we need to know the bottleneck link capacity for the overall connection to be able to produce correct results since this formula gives the throughput over a single link. Second, this formula gives the total number of streams that survive to get this aggregate throughput. However, if we need to know the additional number of streams to open, then we must have an idea about how many other streams are using the link as cross traffic. We must find a way to determine the existing flow number.

2.4 Kola&Vernon Model

A new protocol implementation is given with the purpose of high bottleneck link utilization and low average backlog at the bottleneck link as well as maintaining fairness among all flows utilizing the bandwidth [4]. A target backlog is computed from the current measured bottleneck to achieve the stated goals. Four input parameters are needed for calculations: capacity of the bottleneck link, average and minimum round trip times, and packet loss rate. With these inputs, it is possible to derive average number of flows sharing the bottleneck link:

Step 1 : Calculate backlog of a single flow. $b = S \times d$ where S is the sending rate and $d = RTT_{avg} - RTT_{min}$.

Step 2 : Calculate total backlog . $B = d \times 12/C$ in terms of packets of MTU size 1500 bytes. In this equation C is the capacity of the bottleneck link.

Step 3: Calculate link utilization U.

$$U \approx 2B/(2B+1) \quad (8)$$

Step 4: Calculate average number of equivalent flows n sharing the bottleneck link.

$$n = B/b = (C \times U)/12S \quad (9)$$

If we know the average number of flows sharing the bottleneck link, we can combine this with Altman et al Model. Say that we want a 98% link utilization and we know the capacity of the bottleneck link. We can calculate the optimal n and subtract the average number of flows sharing the link. We find the number of additional streams to open, using:

$$n_{add} = n_{opt} - n \quad (10)$$

However the information for those calculations can be gained in the low level protocol layer. For example the sending rate can only be decided by the underlying protocol.

3 Proposed Models

In this section, we propose several model improvements both based on Hacker et al and Dinda et al models and discuss their derivations and implications.

3.1 Modeling Packet Loss Rate

The shortage of Hacker et al Model is that there is no information on when the point of congestion will occur as a result of opening multiple streams. The reason of that conclusion is that the behavior of the packet loss rate as the number of streams increase is unpredictable. However, if we can find a model to characterize the

packet loss rate, then we can use Equation 2 to calculate the throughput gained by n streams.

To achieve this, a methodology similar to the one in Dinda et al Model can be used. However this time we can not use a partial second order polynomial to model the packet loss rate, since it increases exponentially as the throughput increases logarithmically. By changing the places of Th_n and p_n in Equation 2 we get the following equation:

$$p_n = \frac{MSS^2 c^2 n^2}{RTT^2 Th_n^2} \tag{11}$$

In this case, we define a new variable Th'_n and correlate it to RTT, MSS, n and Th_n. Hacker et al proves that throughput increases linearly in uncongested networks as the number of streams increases. However this is not true for congested networks. The throughput achieved by n streams increases logarithmically:

$$Th'_n = \frac{RTT_n^2 Th_n^2}{MSS^2 c^2} = a' n^{1/x} + b' \tag{12}$$

Ranging x between 2 and a greater number, we determine how sharp the increase in packet loss will be after the point of congestion. By placing Th'_n in Equation 11, we get the following equation for the packet loss of n streams:

$$p_n = \frac{n^2}{Th'_n} \tag{13}$$

Considering we can gather the packet loss rates of transfers with two different stream numbers p_{n_1} and p_{n_2}, we could find the values of a' and b'.

$$a' = \frac{\frac{n_2^2}{p_{n_2}} - \frac{n_1^2}{p_{n_1}}}{n_2^{1/x} - n_1^{1/x}} \tag{14}$$

$$b' = \frac{n_1^2}{p_{n_1}} - a' n_1^{1/x} \tag{15}$$

After finding the value of p_n, we could easily calculate the throughput value of n streams by using Equation 2 in Hacker et al Model. However, this value represents an upper bound on the throughput achieved. In section 6, we show that this calculation gives a better approximation to the throughput curve.

3.2 Increasing Curve Fitting with More Data

The study in [13] shows that the characteristic of a throughput curve increases sharply then becomes stable until it reaches the capacity of the link therefore the model represented fits to the data presented. However in real experimental environment by using actual file transfer protocols(e.g. GridFTP) the results could be different from the proposed situation. Figure 1 presents a file transfer of 512MB with GridFTP over a wide area network using up to 40 parallel streams. As the number of streams increases, the throughput achieved also increases. However, after some point, the created

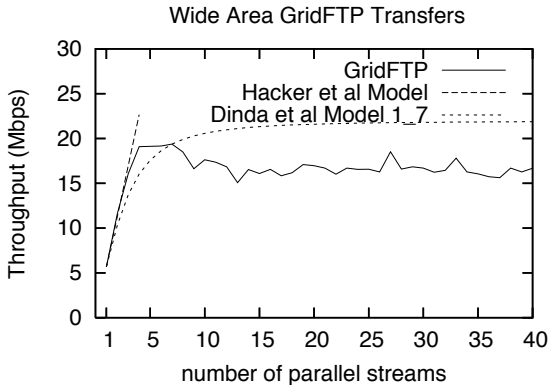

Fig. 1. The aggregate throughput of GridFTP transfers for a 512MB file over 155ms latency wide area network.

congestion by opening too many streams causes a decrease. The peek point in this case gives us the optimum number of streams. The existing models can not predict this behavior. Hacket et al Model predicts the throughput correctly up to the point where the congestion starts and packet loss rate starts to increase. On the other hand Dinda et al Model can predict the throughput behavior correctly up to the point the throughput starts to decrease. However, it can not predict the decreasing part of the throughput curve.

Instead of using two throughput measurements for two different parallelism levels we plan to increase this information level into three measurement values. However, the overhead of gathering extra information must not surpass the actual speed up gained by opening parallel streams. In this case, we may apply two different methodology to make use of the extra information. First, the calculated a' and b' for using parallelism levels n_{12} and n_{13} are averaged either by using arithmetic, geometric or quadratic averaging methods. Throughput can be calculated with the averaged values of a' and b'. Second, as we can see from Figure 1, the throughput curve acts as two different functions. Until reaching the peek point, it acts as a certain function. However, when falling down, it shows a different characteristic. Instead of using a single function, we could break the function into two. By using parallelism level n_1 and n_2, we could model a certain function; and by using n_2 and n_3 we could model the second part. Passing between two functions can be a little sharp; however, this indicates that the optimal number of streams is certainly between n_2 and n_3. The result of experiments are further analyzed in section 6. In the next section, we present a model that smooths out this sharp transition between two functions.

3.3 Logarithmic Modeling of Throughput Curve

The relationship between p, RTT and n is modeled with a partial second order polynomial equation in[13]. However, the proof of their of equation is based on the experiments done. In some of the cases a linear

or Full Second Order polynomial equation may give good results. We know that the packet loss rate increases exponentially. However, we may not know the order of the equation to use. In this case, instead of using a partial second order polynomial, we use an exponential equation whose order may change depending on the a' and b' parameters. The following equation is used to define the variable p'_n we have mentioned before:

$$p'_n = a'e^{b'n} \tag{16}$$

and

$$Th_n = \frac{n}{\sqrt{a'e^{b'n}}} \tag{17}$$

By using two throughput measurements Th_1 and Th_2 with different parallelism levels n_1 and n_2 the following values are calculated for a' and b':

$$a' = \frac{n_1^2}{Th_{n_1}^2 \times e^{b'n_1}} \tag{18}$$

$$b' = \log_{e^{n_1-n_2}} \frac{Th_{n_2}^2}{Th_{n_1}^2} \times \frac{n_1^2}{n_2^2} \tag{19}$$

3.4 Predicting Model Equation Order via Newton's Iteration

The logarithmic modeling of throughput that is explained in the previous section is able to make a smooth transition from the increasing to the decreasing part of the prediction curve. However as the stream number increases the prediction curve approaches to 0 and may not give an approximate prediction result to the actual throughput in the decreasing part of the curve. To be able to make a good prediction we have formulized the p'_n by addition of a new variable to predict the order of the equation as follows:

$$p'_n = a'n^{c'} + b' \tag{20}$$

In this case c' is the unknown order of the equation additional to a' and b'. Our throughput formulation becomes:

$$Th_n = \frac{n}{\sqrt{a'n^{c'} + b'}} \tag{21}$$

To solve this equation, we need three measurements Th_{n_1}, Th_{n_2} and Th_{n_3} on the throughput curve for stream values n_1, n_2 and n_3. Also, c' being to the power n makes the solving of the equation much harder. After several substitutions, we come up with the following equations for a', b' and c':

$$\frac{n_3^{c'} - n_1^{c'}}{n_2^{c'} - n_1^{c'}} = \frac{\frac{n_3^2}{Th_{n_3}^2} - \frac{n_1^2}{Th_{n_1}^2}}{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}} \tag{22}$$

$$a' = \frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2^{c'} - n_1^{c'}} \tag{23}$$

$$b' = \frac{n_1^2}{Th_{n_1}^2} - a'n_1^{c'} \tag{24}$$

The derivations of a' and b' depend on c'. To solve the first equation we applied a mathematical root finding method called Newton's iteration (known also as Newton-Raphson Method). We revised the method to be suitable to our own problem:

$$c'_{x+1} = c'_x - \frac{f(c'_x)}{f'(c'_x)} \tag{25}$$

According to that method, after $x+1$ iterations we are able to find a very close approximation to c'. Starting with a small number for c'_0 we continued to calculate through c'_{x+1}. The value of the most approximate c' depends on only $f(c')$, in this case the first equation above, and its derivative. After calculating a most approximate c' which is possible with only a few iterations, the value of a' and b' can easily be calculated.

3.5 Full Second Order Model

The study in [13] briefly compares the partial second order with linear and Full Second Order models. However the results are compared based on the increasing and then becoming stable characteristics of the throughput and suitable parallelism levels were not used. We believe that a Full Second Order model can predict the increasing and then decreasing characteristics of GridFTP througput as the number of streams increases. In the following sections we also provide a means to determine the best parallelism level samples to fit the models to the actual throughput results.

Regarding to the Full Second Order model, we assume that p'_n is related to a Full Second Order polynomial, other than the partial second order one which was presented before. Adding the linear term to the model will result in a series of changes to the corresponding equations. The following equations are derived to be used in this model.

$$p'_n = p_n \frac{RTT_n^2}{c^2 MSS^2} = a'n^2 + b'n + c' \tag{26}$$

According to Equation 26, we derive:

$$Th_n = \frac{n}{\sqrt{p'_n}} = \frac{n}{\sqrt{a'n^2 + b'n + c'}} \tag{27}$$

In order to obtain the values of a', b' and c' presented in Equation 27, we need the throughput values of three different parallelism levels($Th_{n_1}, Th_{n_2}, Th_{n_3}$) which can be obtained from the predictions of network measurement tools or past data transfers .

$$Th_{n_1} = \frac{n_1}{\sqrt{a'n_1^2 + b'n + c'}} \tag{28}$$

$$Th_{n_2} = \frac{n_2}{\sqrt{a'n_2^2 + b'n + c'}} \tag{29}$$

$$Th_{n_3} = \frac{n_3}{\sqrt{a'n_3^2 + b'n + c'}} \tag{30}$$

By solving the following three equations we could place the a',b' and c' variables to Equation 27 to calculate the throughput of any parallelism level. Based on equations 31,32 and 33, the values of a',b' and c' can be calculated easily.

$$a' = \frac{\frac{\frac{n_3^2}{Th_{n_3}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_3 - n_1} - \frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2 - n_1}}{n_3 - n_2} \tag{31}$$

$$b' = \frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2 - n_1} - (n_1 + n_2)a' \tag{32}$$

$$c' = \frac{n_1^2}{Th_{n_1}^2} - n_1^2 a' - n_1 b' \tag{33}$$

4 Algorithm to Find Best Parallelism Levels to Calculate Throughput

Each model presented above can show their best performances if the sample throughput data to calculate the predicted throughput can be chosen from appropriate parallelism levels. All of the models either need 2 or 3 throughput data of different parallelism levels. Hence there exists many combinations if we have more than three pairs(n, Th_n) of data and it is important for us to find the best combination to minimize the distance between historical or prediction data and calculated throughput of n streams based on the models presented. In this section, we introduce an algorithm about how to find the appropriate combination of pairs. The algorithm can both be used for evaluation of different prediction models presented as well as optimization in choosing the best historical or prediction data.

The outline of the algorithm is given below for the models that need three sample throughput data pairs. The algorithm can easily be adjusted for models requiring only two throughput data pairs with a few changes. We only need to leave out one of the loops and calculation of c'.

Input: $CollectiveData[m][2] = \begin{pmatrix} n_0 & Th_{n_0} \\ n_1 & Th_{n_1} \\ \cdots & \cdots \\ n_l & Th_{n_l} \\ \cdots & \cdots \\ n_{m-1} & Th_{n_{m-1}} \end{pmatrix}$

Output: $(n_r, Th_{n_r})(n_s, Th_{n_s})(n_t, Th_{n_t})(a', b', c')err_{min}$

Begin
  for $i \leftarrow 0$ **to** $m-2$ **do**
    for $j \leftarrow i+1$ **to** $m-1$ **do**
      for $k \leftarrow j+1$ **to** m **do**
        $a, b, c \leftarrow$ GETCOEFFICIENT($CollectiveData[i]$, $CollectiveData[j], CollectiveData[k]$)
        $err \leftarrow$ ERREVALUATION($a, b, c, CollectiveData$)
        if $minerr$ *is not initialized* **then**
          $minerr \leftarrow err$
          $r, s, t \leftarrow i, j, k$
          $a', b', c' \leftarrow a, b, c$
        else if $err < minerr$ **then**
          $minerr \leftarrow err$
          $r, s, t \leftarrow i, j, k$
          $a', b', c' \leftarrow a, b, c$
        end if
        $k \leftarrow k+1$
      end for
      $j \leftarrow j+1$
    end for
    $i \leftarrow i+1$
  end for
  $n_r \leftarrow CollectiveData[r][0]$
  $n_s \leftarrow CollectiveData[s][0]$
  $n_t \leftarrow CollectiveData[t][0]$
  $Th_{n_r} \leftarrow CollectiveData[r][1]$
  $Th_{n_s} \leftarrow CollectiveData[s][1]$
  $Th_{n_t} \leftarrow CollectiveData[t][1]$
End

The input of the algorithm is a $m \times 2$ array which can be obtained from a file which contains all the data pairs that are collected from past GridFTP transfers or it can also be obtained instantly from network performance measurement tools. In each pair (n_l, Th_{n_l}), n_l stands for the number of streams, correspondingly, Th_{n_l} stands for the throughput with respect to n_l.

The output of the algorithm is as follows:

- (n_r, Th_{n_r}) (n_s, Th_{n_s}) (n_t, Th_{n_t}) stand for the streams and the corresponding throughputs when the prediction function best fits the historical or prediction data.
- (a', b', c') stands for the polynomial coefficients when the prediction function best fits the historical or prediction data.
- $errmin$ stands for the minimum average error of all the trials to find the polynomial coefficients. The fact is that err approaches its miminum value $errmin$ when the prediction function best fits the actual throughput.

The functions are defined as:

- $CollectiveData$ stands for all the data in the $m \times 2$ array; $CollectiveData[i]$ stands for the i_{th} line of data, i.e., $CollectiveData[i][0]$ and $CollectiveData[i][1]$.
- $GetCoefficient(\cdots)$ will use three pairs of data as its parameters and return the value of the polynomial coefficients in the prediction function.
- $ErrEvaluation(\cdots)$ will use the parameters presented above to calculate err value based on a certain criteria such as mean square error. The details of the function will be discussed in detail in the following sections.

In our algorithm, we have considered to minimize the average distance between the historical or tool prediction data and calculated predicted throughput values of all parallelism levels based on the models. Let ε denote the distance between historical or tool predicted throughput and calculated throughput of models, $Th^a_{n_i}$ denote the historical or tool predicted throughput value of parallelism level n_i and $Th(n_i)$ denote the calculated throughput of parallelism level n_i based on the models. The error value based on this distance (Err) is calculated in Equations 34 and 35.

$$\varepsilon_i = Th^a_{n_i} - Th(n_i) \tag{34}$$

$$\begin{aligned} Err &= \sqrt{\frac{\varepsilon_0^2 + \varepsilon_1^2 + \cdots + \varepsilon_{m-1}^2}{m}} \\ &= \sqrt{\frac{\sum_{i=0}^{m-1}(Th^a_{n_i} - Th(n_i))^2}{m}} \end{aligned} \tag{35}$$

5 Delimitation of Coefficients

The prediction by using some combinations of streams causes the error rate to be very large and the prediction curve not to show the characteristics of the historical or tool predicted throughput curve. The difficulty is that we can not know if the prediction is successful or not before the error rate is calculated and this results in a large number of comparisons in our algorithm. However, if we can predict the accuracy of the prediction curve using a certain combination of streams based on the delimitation of the coefficients (a',b' and c') without comparing the predicted throughput with the historical or tool prediction data, then we can improve the efficiency of the algorithm significantly. For this purpose, we introduce propositions for all models and prove it in the following subsections.

5.1 Propositions

The following inequalities represents the range of values that a certain coefficient should take in our models to reflect the characteristics of the throughput curve better.

i)Dinda et al model:

1) $a' > 0$
2) $b' > 0$

ii)Newton's Iteration Model:

1) $a' > 0$
2) $b' > 0$
3) $c' \geq 2$
4) $\left(\frac{2b'}{a'(c'-2)}\right)^{\frac{1}{c'}} > 1$

iii)Full Second Order Model:

1) $a' > 0$
2) $b' < 0$
3) $c' > 0$
4) $2c' + b' > 0$

iv)The Averaging and Break function methods uses the same requirements as those of Dinda et al Model.

5.2 Proof of Dinda et al Model Coefficients Requirements

In order to get a monotonically increasing throughput function in which the throughput increases logarithmically as the number of streams increases and becomes stable with an upper bound, we should guarantee that the derivative of the throughput function must be greater than 0.

$$\begin{aligned} Th'_{din} &= \frac{b'}{(a'n^2 + b')^{\frac{3}{2}}} > 0 \\ &\Rightarrow \begin{cases} b' > 0 \\ a'n^2 + b' > 0 \end{cases} \quad \forall n \in N^+, n \leq optnum \\ &\Rightarrow \begin{cases} a' > 0 \\ b' > 0 \end{cases} \end{aligned}$$

When we take the limit of the throughput function we can see that it increases monotonically with an upper bound $\frac{\sqrt{a'}}{a'}$ as well.

$$\begin{aligned} \lim_{n\to\infty} \frac{n}{\sqrt{a'n^2 + b'}} &= \lim_{n\to\infty} \frac{\sqrt{a'n^2 + b'}}{a'n} \\ &= \lim_{n\to\infty} \frac{\sqrt{a' + \frac{b'}{n^2}}}{a'} \\ &= \frac{\sqrt{a'}}{a'} \end{aligned} \tag{36}$$

5.3 Proof of Newton's Iteration Model Coefficients Requirements

Newton's Iteration Model should give us a prediction curve that increases first to a peak value as the number of streams increases, and then decreases gradually to a lower bound.

Based on Equation 21, we know that $c' \geq 2$, otherwise, the limit of the throughput function will be infinity. When $c' = 2$, the throughput equation will be the

same as the Dinda et al Model. From the discussion about Dinda et al Model, we also know the throughput function of that model increases monotonically. In this case, in order to get a throughput prediction curve with the characteristics described in the first paragraph, c' must be greater than 2. Also we must ensure that $a' > 0$, otherwise, the term inside the square root will be less than zero.

To meet the increasing and decreasing properties of the throughput curve, the first derivative should be positive initially, at the peek point it should become zero, and finally become negative. Since the denominator is positive, we only need to control the numerator of the throughput function. If $b' \leq 0$, the numerator will be negative, thus we conclude that $b' > 0$.

When we equalize the numerator of the derivative function to 0, we find that $n = \left(\frac{2b'}{a'(c'-2)}\right)^{\frac{1}{c'}}$. Since the throughput will increase along with the stream number, the optimum value of n should be greater than 1. Thus we get an inequality of $\left(\frac{2b'}{a'(c'-2)}\right)^{\frac{1}{c'}} > 1$.

5.4 Proof of Full Second Order Model Coefficients Requirements

The proof for the Full Second Order Model is similar to the Dinda et al Model except that we have to break the derivative of the function into 3 parts. In order to get a curve which increases first and then decreases monotonically, we should guarantee that the first derivative of the throughput function is positive for the increasing part, and later is zero when it reaches the peak, and finally is negative for the decreasing part of the curve.

$$
\begin{aligned}
Th'_{ful} &= \begin{cases} \frac{\frac{b'}{2}n+c'}{(a'n^2+b'n+c')^{\frac{3}{2}}} > 0 & n < optnum \\ \frac{\frac{b'}{2}n+c'}{(a'n^2+b'n+c')^{\frac{3}{2}}} = 0 & n = optnum \\ \frac{\frac{b'}{2}n+c'}{(a'n^2+b'n+c')^{\frac{3}{2}}} < 0 & n > optnum \end{cases} \\
&\Rightarrow \begin{cases} \frac{b'}{2}n + c' > 0 & n < optnum \\ \frac{b'}{2}n + c' = 0 & n = optnum \\ \frac{b'}{2}n + c' < 0 & n > optnum \\ a'n^2 + b'n + c' > 0 & \forall n \in N^+ \end{cases} \\
&\Rightarrow \begin{cases} a' > 0 \\ b' < 0 \\ c' > 0 \\ optnum = \frac{-2c'}{b'} > 1 \end{cases} \\
&\Rightarrow \begin{cases} a' > 0 \\ b' < 0 \\ c' > 0 \\ 2c' + b' > 1 \end{cases}
\end{aligned}
$$

When the limit is calculated we could see that *Full second order* Model increases first, then reaches a peak value, later decreases to a lower bound $\frac{\sqrt{a'}}{a'}$.

$$
\begin{aligned}
\lim_{n\to\infty} \frac{n}{\sqrt{a'n^2+b'n+c'}} &= \lim_{n\to\infty} \frac{2\sqrt{a'n^2+b'n+c'}}{2a'n+b'} \\
&= \lim_{n\to\infty} \frac{\sqrt{a'+\frac{b'}{n}+\frac{c'}{n^2}}}{a'} \\
&= \frac{\sqrt{a'}}{a'} \qquad (37)
\end{aligned}
$$

The predictability of the coefficients for the models presented above is of great significance. When we get the value of the coefficients through a certain combination of some streams, we can use the above propositions to determine whether they are acceptable or not. If the coefficients do not hold the propositions, we do not calculate the throughput for those combinations of parallelism levels and that saves us a lot of time, since we can decide if the specified combination of data can give us a correct prediction result based on a few simple proposition checks without doing a significant amount of calculations and comparisons.

6 Experimental Analysis

In this section, we present our wide area test results regarding GridFTP transfers and give prediction results of several methods used. The first wide area test bed consists of two Linux machines, one is a Redhat workstation in the Center for Computation and Technology at Lousiana State University, the other belongs to a Suse cluster in University of Trento in Italy. Both machines have Globus Toolkit [15] installed. Also for packet behavior measurements, we used a Linux network analyzer tool called Wireshark [16]. We have conducted GridFTP transfers with a file size of 512MB. The number of streams ranged between 1 and 40. The results presented in Figure 1 are the average of multi-iteration of tests. The actual file transfers by means of a protocol, in our case GridFTP, follows a different characteristics than presented in study [13]. In this case, the aggregate throughput achieved by parallel streams increases as the number of streams increases, after reaching its peek it presents a wavy behavior, however further increasing of stream number results in a descent in throughput. The current models can not predict this unstable behavior. To prove the correctness of our models we also conducted the same tests with different parameter settings in the LONI network [17] by using two IBM clusters of which results will be represented in the following subsections.

6.1 Comparison of Models with Randomly Chosen Paralellism Levels

Starting with the shortcomings of Hacker et al Model, the idea of the total aggregate throughput of parallel streams being equal to the throughput of a single stream times the number of streams seems to be proven in uncongested networks. Figure 1 shows that up to 3

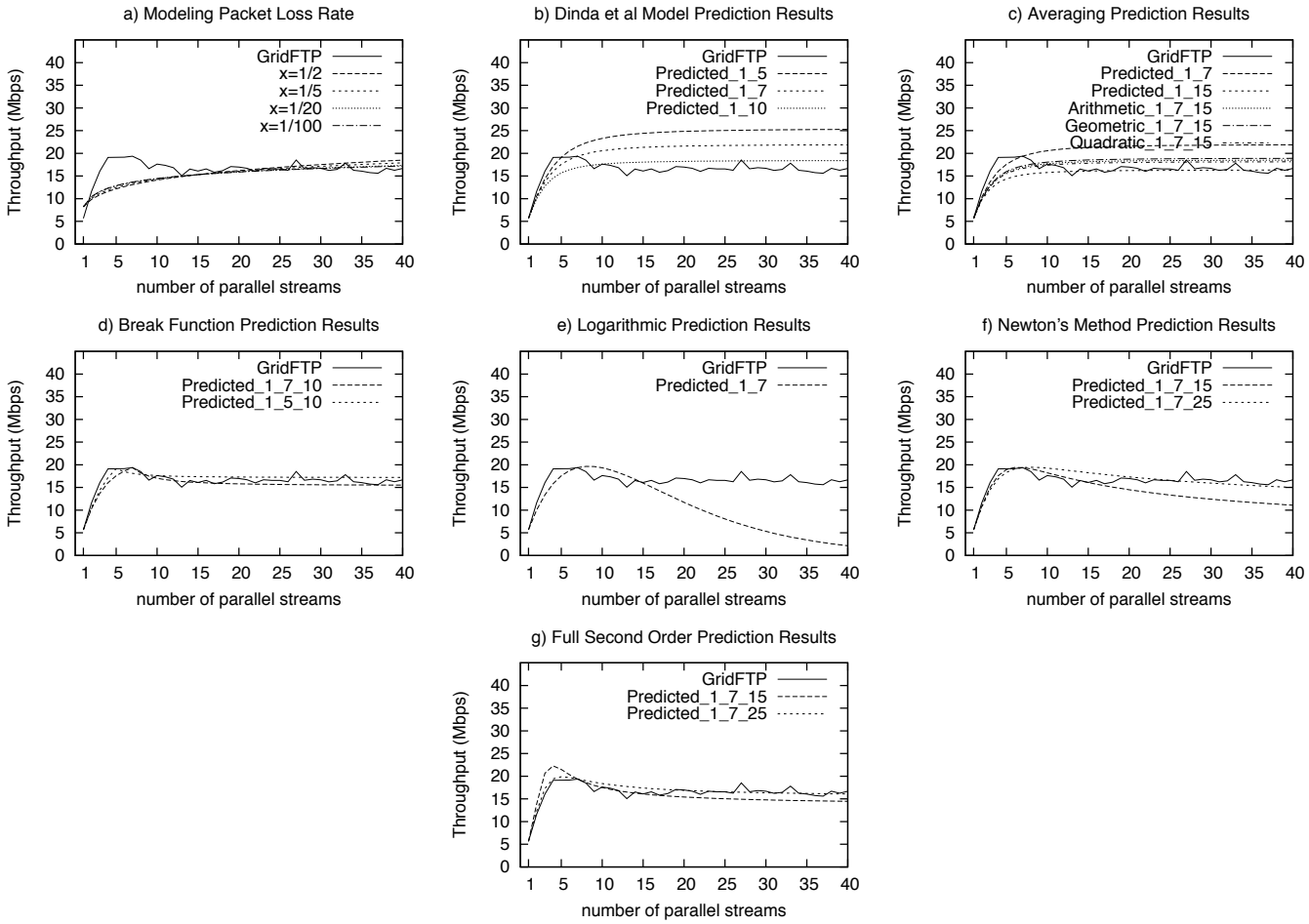

Fig. 2. The prediction results of GridFTP transfers by applying Dinda et al Model and its improvements.

streams this equivalence is true more or less, however after that point the linear ascent of throughput curve turns into a logarithmic one indicating the existence of congestion in the network. The primary reason of this behavior is the change in packet loss rate due to congestion. By modeling packet loss according to some measured packet loss rates of different parallelism levels taken from Wireshark with our presented model and applying Equation 11 we get a predicted packet loss rate and replace it in Equation 2 to calculate throughput. As a result, we get the following results presented in Figure 2.a. The difficulty of applying this model is that we need a lot of information like MSS, RTT and packet loss rate. The figure shows that we can get a logarithmic increase which better suits assuming that packet loss rate increases exponentially as we have proposed in our model. By increasing the x value we could get a sharper knee and a flatter continuous behavior for the rest of the stream numbers.

We also implemented Dinda et al Model and all of the improvements based upon it for randomly chosen parallelism level information. Figure 2.b represents the prediction results of Dinda et al Model for different parallelism levels. The prediction curve calculated with throughput results of 1 and 5 parallel streams presents higher results than the ones calculated with 1 and 7, and also 1 and 10. All of the prediction curves have a logarithmic ascent behavior and then becomes stable for high number of streams but never fall down. The study in [13] mentions that best parallelism values to use are either 1-5 or 1-10. However we could only tell this after some experiences with the transfers. Instead of that we could try to increase our information level by using 3 parallelism levels. According to our approach, we may detect the behavior change in the function by calculating average a' and b' values. The results of this model is given in Figure 2.c. The averaging results give that best stream number values to predict this curve is between 7 and 10. However we do not need to know exactly this number by using our averaging approach. The averaged curve gives a closer result to both the increasing and decreasing part of the GridFTP throughput curve.

Figure 2.d shows the prediction results taken by 1-5-10 and 1-7-10 parallelism levels for Break Function model. We could see that the prediction curve almost perfectly suits the GridFTP curve. Although the transitions are a little sharp between the pieces of the curve that gives us an exact idea where the optimal parallel stream number

lies. The result of the logarithmic model is shown in Figure 2.e. The transition between the ascending and descending part of the throughput curve is smoothed out. However, the descending part of the throughput curve can not be predicted well as the curve approaches to 0 as the number of streams increases further. We have solved the problems of both the 'break function' and 'logarithmic prediction' methods by predicting model equation order dynamically. Figure 2.f shows the results of the predicted throughput by using Newton's Iteration mentioned in Section 3.4. The best results are taken with 1, 7 and 25 parallelism levels although 1, 7 and 15 give pretty close results. We are able to predict the actual GridFTP curve with a smooth transition between the increasing and decresing part of the curve and with a good approximation overall. In the last model, we have applied a Full Second Order equation and have seen that if the correct parallelism levels are used this model predicts the throughput value precisely. Hence, although 1-7-15 gives exaggerated results for the maximum point in throughput, 1-7-25 gives accurate results. In this case, it becomes an important decision which paralelism levels to use to get the most accurate results.

6.2 Effect of Using Randomly Chosen Datasets

To evaluate the performance of different models by using some specified data set may not give us a good approximation of the power of the models. In existence of the appropriate combination of data sets, a model may give less error rates. However, for different sets of data, while some models give accurate results the others may not. In the previous experiments we assumed that we have the total throughput values of all parallelism levels, in our case, up to 40 streams. However, in practical cases, it might not be possible to gather the whole data set information. The problem is whether we can use the limited number of data to find the suitable coefficients for the prediction models. In this section, we design an evaluation strategy which will use randomized datasets that will cover subsets of all the parallelism level information. Based on this strategy we also define new metrics and evaluate our models based on them.

Since the number of data used for model evaluation will affect the accuracy of our prediction, we will try different numbers of data in the evaluation procedure. We use a data file that keeps all the data we get from the experiments we have conducted. We call all the data in this file which includes the throughput values of all parallelism levels up to 40 the *population*, from which we get *sample*s with different numbers of data for our evaluation procedure. The sample data sets are taken from the population randomly.

6.2.1 Comparison Metrics

For each sample data set, we calculate the coefficients and get the equation to predict the throughput. The coefficients are obtained from the sample data set, however, we will make the comparison based on both the data from the *population* and the data from the *sample*.

The predicted throughput can not be calculated for some paralelism levels with the coefficients derived from a specified stream combination selection as they will make the term of the square root less than zero or make the denominator equal to zero. So we call those stream numbers out of the domain. We say that the *Predictability* of a specified equation is poor if the number of streams out of the domain is large in proportion. We use $P_{pre[m\_i]}$ to stand for the predictability of a specified equation derived from the data set named m_i where m represents the number of data and i represents the ith set. Let the number of streams within domain be represented by N_{ind} and the number of streams out of domain be represented by N_{outd}, then we define $P_{pre[m\_i]}$ as follows:

$$P_{pre[m\_i]} = \frac{N_{ind}}{N_{outd} + N_{ind}} \tag{38}$$

The equation solved by some combinations of parallelism level data will cause the predictability to be in a poor situation, while some other combinations may cause the predictability to be stronger. If there are lots of combinations of data which make the predictability to be in a poor situation, then we say that this prediction model is sensitive to data. Let $P_{sen[m\_i]}$ to be denoted as the sensitivity of the sample data set, the number of combinations which make $P_{pre[m\_i]} = 1$ to be denoted as N_{eff} and the number of all the combinations of a sample data to be denoted as N_{sam}, then we calculate $P_{pre[m\_i]}$ as follows:

$$P_{sen[m\_i]} = \frac{N_{eff}}{N_{sam}} \tag{39}$$

For evaluation of accuracy of the models, we use the quadratic average of the distance metric that was used in BFC Algorithm for stream values within the domain.

$$Err_{[m\_i]} = \sqrt{\frac{\sum_{k=1}^{N_{ind}} \epsilon_k^2}{N_{ind}}} \tag{40}$$

We propose to find a best fitted combination of data for the data set m_i such that $P_{pre[m\_i]} = 1$ and $Err_{[m\_i]}$ gets the minimum value compared with all possible combinations for the sample data. The following equation represents the best error rate value among different combinations of parallelism levels.

$$Err_{best[m\_i]} = min_{N_{eff}} \left(Err_{[m\_i]} \right) \tag{41}$$

Finally, by taking the average of results for all random data sets, we compare our models. In the following equations k_m represents the number of data sets of m data.

$$P_{sen[m]ave} = \frac{\sum_{i=0}^{k_m - 1} P_{sen[m\_i]}}{k_m} \tag{42}$$

$$P_{pre[m]ave} = \frac{\sum_{i=0}^{k_m - 1} P_{pre[m\_i]}}{k_m} \tag{43}$$

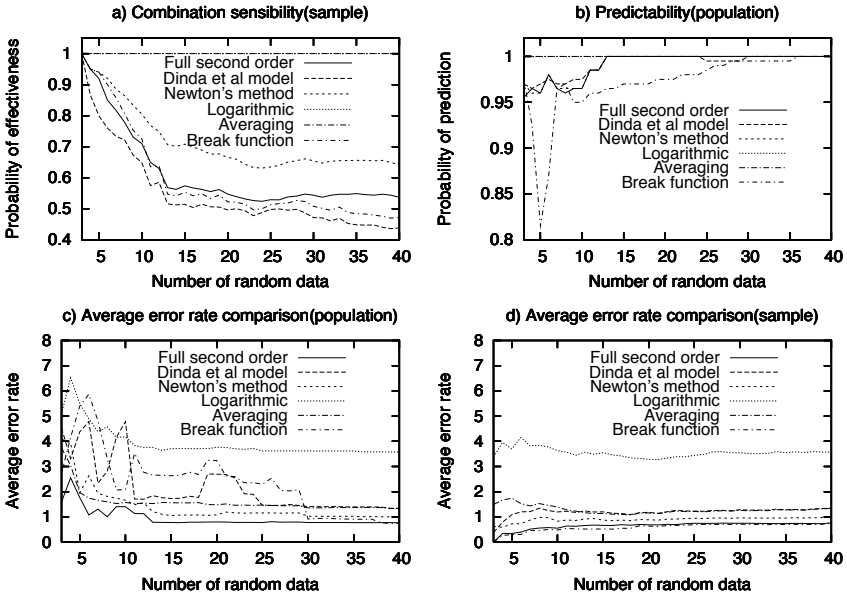

Fig. 3. Model Evaluation based on predictability, sensibility and error rate.

$$Err_{best[m]ave} = \frac{\sum_{i=0}^{k_m-1} Err_{best[m\_i]}}{k_m} \quad (44)$$

6.2.2 Evaluation

In this section, we present the results based on the evaluation metrics we defined above. Figure 3.a shows the sensibility of data that is used for prediction using all models as the number of data is increased in the samples. We select one combination of parallelism level data each time to get the coefficients of the throughput equation. The y-axis is the probability of the effectiveness of these combinations. If the equation derived from a specified combination can predict the throughput of each number of stream, i.e., $P_{pre[m\_i]} = 1$, we say this combination is effective in the data set m_i. The sensibility is inversely related to the effectiveness of the combinations. From Figure 3.a we can see that the logarithmic and the averaging model is not sensible to the data. So they are the most stable models. Newton's Iteration follows them with a high probability of effectiveness. Full second order, break function and Dinda et al. models are the most sensible to data. Another observation is that as the number of data increases, the combinations become more sensitive to data.

Figure 3.b presents the predictability of a throughput equation with specified coefficients. Once the equation is applied with the coefficients, the throughput value for some number of streams can not be calculated as the denominator becomes equal to zero or the term inside the square root is less than zero. In this case, we say that those stream numbers are out of the domain. We can not predict the throughput of a stream number if it is out of the domain. From the figure, it can be seen that logarithmic and averaging models are the most predictable while Newton's Iteration Model, Dinda et al Model and Full Second Order Model can only reach them for large number of data. The worst performance is with the break function model.

Figure 3.c presents the average error for the population data, in which case the predicted throughput is calculated for every parallelism level and compared to the whole data set of experimental results. According to the figure, the best results are taken with Full Second Order and Newton's Iteration and the worst ones are taken with logarithmic model.

When the predicted results are calculated for only the parallelism level existing in the sample, break function model gives the best results and Full Second Order and Newton's Iteration follows it (Figure 3.d). The worst are again taken with logarithmic model. The results are similar when comparison is made for population and sample data sets. This is a strong proof of the correctness of our scheme of using a subset of the pupulation data to predict the throughput of the population.

6.3 Decreasing DataSet Size with Intelligent Selection of Parallelism Levels

In the previous evaluation sections, we have presented the power of the models for randomly chosen parallelism levels and randomly chosen data sets. When we choose a random combination of parallelism levels, it is a high possibility that the chosen levels may not reflect the characteristics of the throughput curve. The algorithm we have presented solves this problem by finding the best coefficients. However it tries a large number of combinations to reach a steady result and it may need a certain size of historical data set to give a good prediction hence can not be used with an online strategy which uses the prediction results of tools rather than past historical transfers.

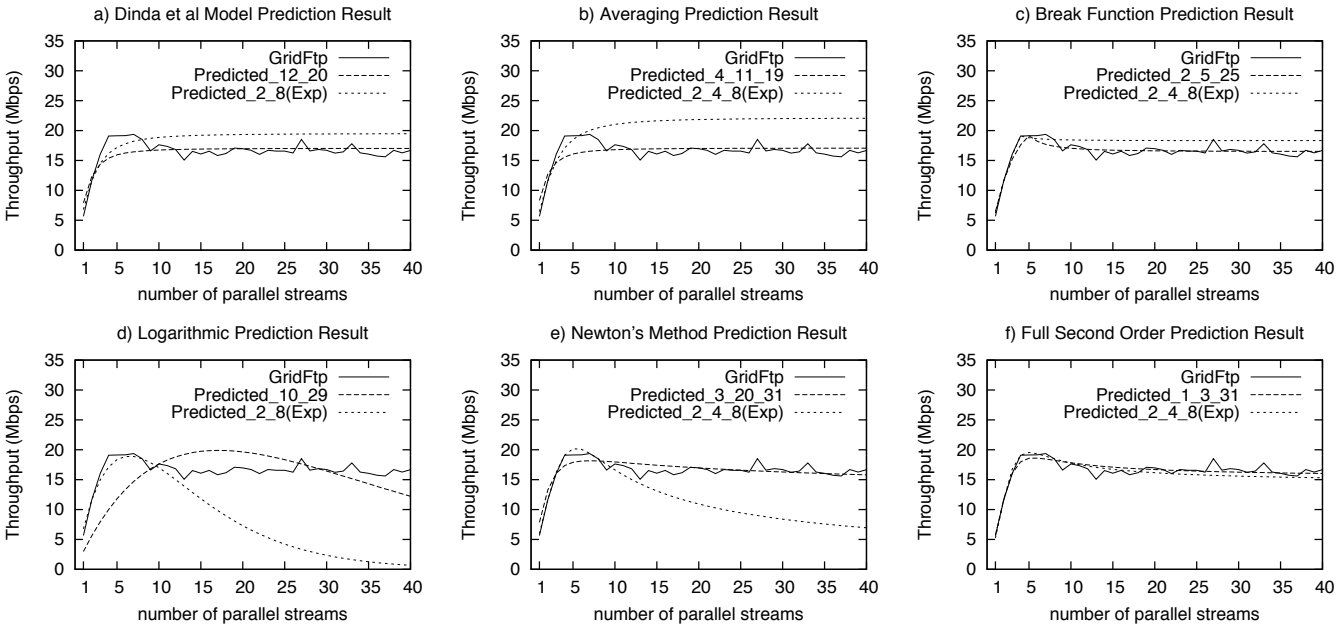

Fig. 4. Prediction of peak point with intelligent parallelism level selection.

In this section, we provide an intelligent selection strategy which decides on less number of data and can be used with an online model as well. Our previous experiences showed that it is better if we choose the parallelism levels not close to each other. So we applied an exponential increase strategy by selecting the stream numbers that are power of 2: $1, 2, 2^2, 2^3, ..., 2^k$. Each time we double the number of streams until the throughput starts to drop down or increase very slowly compared to the previous level. After k+1 steps we gather k+1 parallelism level throughput data and apply the Best-fitted Coefficient Algorithm to find the best parallelism levels. In this case we only need a data size of O(log 2^k) and hence do less number of comparisons.

Figure 4 shows the difference between the basic Best-fitted Coefficient Algorithm results and the exponential selection strategy for all the models. This strategy focuses on better prediction of the optimal parallelism level which is the peak point of the curves. While the basic algorithm gives better results for the average throughput distances in most of the cases, with exponential increase strategy we are able to better predict the peak throughput point hence the number of streams that gives it. More importantly, we could find this point with a small size dataset of exponentially increasing points. According to the figure, only 4 points (1,2,4 and 8) seems to be enough to get a good prediction curve. Best results are taken with Full Second Order model in terms of both throughput distance and peak point prediction. Newton's Iteration and Logarithmic models can predict the peak point well, however with a trade of high throughput distance between GridFTP and prediction results.

6.4 LONI Experiments

To prove the correctness of models and algorithms better, we set up another testbed in the LONI network. Two IBM clusters with 10Gbps network interfaces and 512KB buffer sizes are used. Although the underlying network has the capacity of more than 10Gbps, the end-systems are capable of transferring in much less speeds and the network transfer speed varies a lot. For that experiment, we used a much bigger file size of 5GB. Again, the number of parallel streams is ranged between [1-40]. Depending on the different characteristics of the network, the fall after reaching the peak point of achievable throughput was much steeper than the previous experiments. Figure 5 shows the application of the Best-fitted Coefficients algorithm and the exponential selection strategy on the GridFTP transfer curve. While both algorithms can not predict the peak point well for Dinda et al and Averaging models, the exponential selection strategy can predict better for Break Function, Logarithmic, Newton's Iteration and Full Second Order models. The best results are taken by applying the Newton's Iteration Model with the exponential selection strategy while Full Second Order and Break Function models follow it. Although the Logarithmic model can not predict the exact peak point, it gives a very close approximation.

In the end, we solve the problems regarding the following cases:

- A good prediction could be made with limited number of data.
- A practical application could be realized by using online prediction tools instead of using past data transfers.

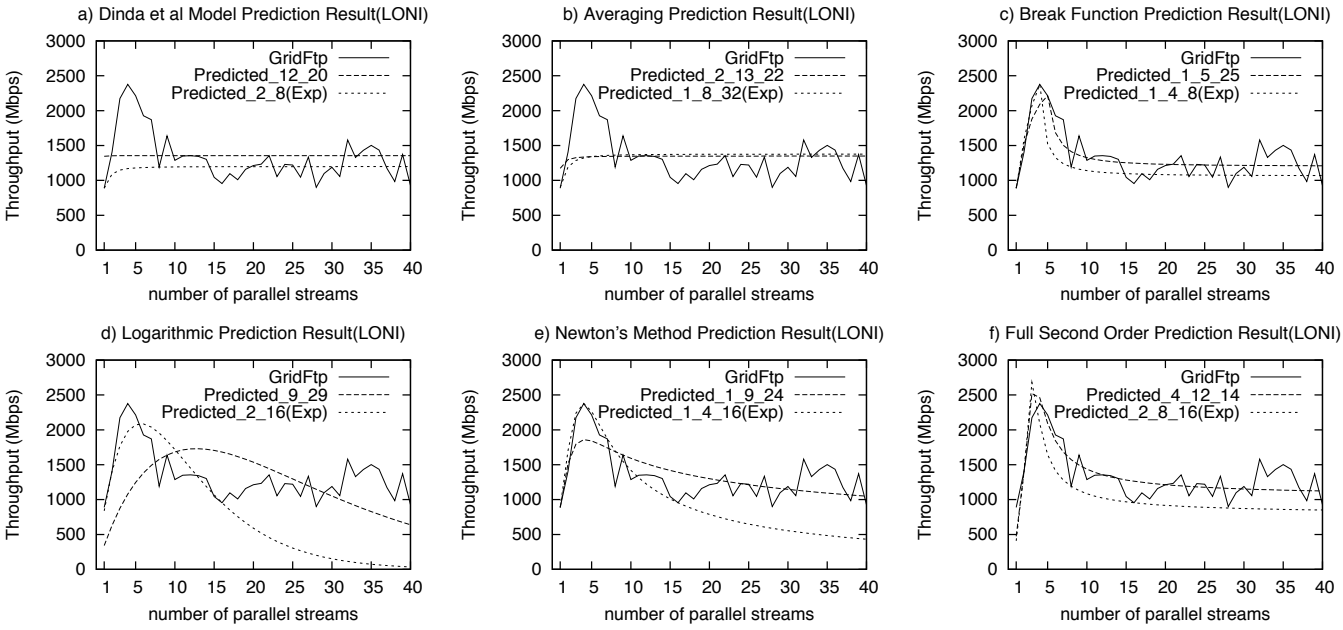

Fig. 5. Comparison of models and algorithms in the LONI network.

- The correct set of parallelism levels could be selected with an intelligent method which has negligible computation overhead.

7 Optimization Models Implementation on Applications

All data-intensive grid applications need high performance for their data transfers. However there are some applications such as data-aware schedulers (e.g. Stork [6]) and storage resource planners (e.g. SRM [18]) that serve to enable the other applications and try to reach their performance requirements. Optimization models we have presented in this study could provide these systems with a powerful tool and shorten their job run time. We have implemented a prototype of our models as an optimization service in the Stork scheduler [6] which enables the user with the option to optimize their data transfer jobs. From that point on, the service does samplings if there are no available history information to apply the models and if there is, the service applies the models on results of the previous transfers. Based on the optimized parameters for parallel streams, the scheduler performs the transfer job.

We have submitted data transfer jobs to Stork scheduler to be performed between two clusters named QueenBee and Oliver with 1Gbps interfaces at LONI network [17]. Figure 6 presents the averaged results of job turnaround times with or without applying the optimization models. Full Second Order Model is applied for this experiment. The time for optimized transfers also includes the time of sampling transfers to gather optimization data as well as the model calculations. The file sizes are ranged between 100MB to 6GB. Based on the results, the optimized data transfer times are much less than the non-optimized data transfers even if no history information is used and the model data is gathered online. It shows us that our models can even be applied online and still improve the data transfer time.

8 Conclusion

The parallel transfer behavior of a TCP-based protocol, GridFTP, over wide area networks is analyzed and several prediction models are presented and improved according to the characteristics of the transfers. It has been observed that the aggregate throughput starts to fall down in existence of congestion and none of the models could mimic this behavior. By using minimum information on historical results, we have improved the current models to predict the throughput curve of GridFTP and we have observed promising results. Theoretically we prove that our improved models can minimize the average error compared with the existing models. Furthermore, we propose assumptions about the delimitation of the coefficients for our improved model and prove it mathematically. After the mathematical analysis and proof of the correctness regarding to our assumptions, we design an algorithm to find out the coefficients of the throughput function. Also we present an intelligent strategy to choose appropriate parallelism levels and decrease data set size. A detailed experimental analysis is done to support our ideas using multiple sampling, enumerating and averaging. Based on the results of our experiments we conclude that we are able to predict the throughput behavior of parallel transfers with Full Second Order and Newton's Iteration with very good accuracy and with a limited data size of historical transfers.

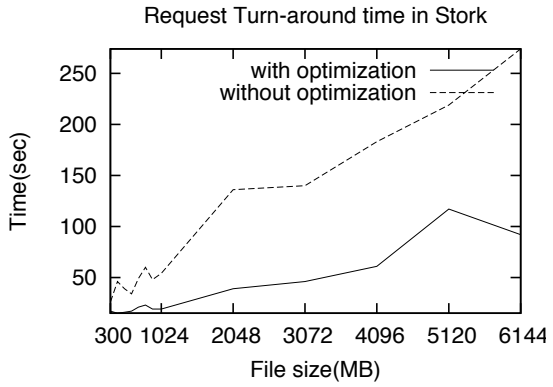

Fig. 6. Time comparison in Stork with and without optimization from qb1.loni.org to oliver1.loni.org.

ACKNOWLEDGMENTS

This project is in part sponsored by the National Science Foundation under award numbers CNS-0846052 (CAREER), CNS-0619843 (PetaShare), OCI-0926701 (Stork) and EPS-0701491 (CyberTools), by the U.S. Department of Energy under Award Number DE-FG02-04ER46136 (UCoMS), and by the Board of Regents, State of Louisiana, under Contract Numbers DOE/LEQSF (2004-07), NSF/LEQSF (2007-10)-CyberRII-01, and LEQSF(2007-12)-ENH-PKSFI-PRS-03.

REFERENCES

[1] L. Eggert, J. Heideman, and J. Touch, "Effects of ensemble tcp," *ACM Computer Communication Review*, vol. 30(1), pp. 15–29, Jan. 2000.
[2] R. P. Karrer, J. Park, and J. Kim, "Adaptive data block scheduling for parallel streams," Deutsche Telekom Laboratories, Tech. Rep., 2006.
[3] J. Crowcroft and P. Oechslin, "Differentiated end-to-end internet services using a weighted proportional fair sharing tcp," *ACM SIGCOMM Computer Communication Review*, vol. 28(3), pp. 53–69, Jul. 1998.
[4] G. Kola and M. K. Vernon, "Target bandwidth sharing using endhost measures," *Performance Evaluation*, vol. 64(9-12), pp. 948–964, Oct. 2007.
[5] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke, "Applied techniques for high bandwidth data transfers across wide area networks," in *Proc. International Conference on Computing in High Energy and Nuclear Physics (CHEP01)*, Beijing, China, Sep. 2001.
[6] T. Kosar and M. Livny, "Stork: Making data placement a first class citizen in the grid," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS04)*, 2004, pp. 342–349.
[7] G. Kola, T. Kosar, and M. Livny, "Run-time adaptation of grid data-placement jobs," *Scalable Computing: Practice and Experience*, vol. 6(3), pp. 33–43, 2005.
[8] H. Sivakumar, S. Bailey, and R. L. Grossman, "Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *Proc. IEEE Super Computing Conference (SC00)*, Texas, USA, Nov. 2000, pp. 63–63.
[9] H. Balakrishman, V. N. Padmanabhan, S. Seshan, and R. H. K. M. Stemm, "Tcp behavior of a busy internet server: Analysis and improvements," in *Proc. IEEE Conference on Computer Communications (INFOCOM98)*, California, USA, Mar. 1998, pp. 252–262.
[10] T. J. Hacker, B. D. Noble, and B. D. Atley, "Adaptive data block scheduling for parallel streams," in *Proc. IEEE International Symposium on High Performance Distributed Computing (HPDC05)*, Jul. 2005, pp. 265–275.
[11] D. Lu, Y. Qiao, and P. A. Dinda, "Characterizing and predicting tcp throughput on the wide area network," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS05)*, Jun. 2005, pp. 414–424.
[12] T. J. Hacker, B. D. Noble, and B. D. Atley, "The end-to-end performance effects of parallel tcp sockets on a lossy wide area network," in *Proc. IEEE International Symposium on Parallel and Distributed Processing (IPDPS02)*, 2002, pp. 434–443.
[13] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante, "Modeling and taming parallel tcp on the wide area network," in *Proc. IEEE International Symposium on Parallel and Distributed Processing (IPDPS05)*, Apr. 2005, p. 68b.
[14] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic, "Parallel tcp sockets: Simple model, throughput and validation," in *Proc. IEEE Conference on Computer Communications (INFOCOM06)*, Apr. 2006, pp. 1–12.
[15] Globus toolkit. [Online]. Available: http://www.globus.org
[16] Wireshark. [Online]. Available: http://www.wireshark.org
[17] Loni optical network. [Online]. Available: http://www.loni.org
[18] Srm. [Online]. Available: https://sdm.lbl.gov/srm-wg/

Esma Yildirim has received her BS degree from Fatih university and MS degree from Marmara University Computer Engineering Departments in Istanbul, Turkey. She worked as a Lecturer in Fatih University Vocational School until 2006. She is currently in Louisiana State University as a Phd student in Computer Science since August 2006. She is working for CCT as a graduate research asistant. Her research interests are Data-intensive Distributed computing and High-performance Computing.

Dengpan Yin has received his Master's degree of computer science from Southern University in May 2008. Currently, he is pursuing his Phd of computer science at Louisiana State University. Meanwhile, he is working as a graduate research assistant at CCT of LSU. His research interests include distributed computing, Grid computing, parallel computing and high performance networks.

Tevfik Kosar is an Assistant Professor in the Department of Computer Science and in the Center for Computation & Technology (CCT) at LSU since Fall 2005. He holds a B.S. degree in Computer Engineering from Bogazici University, Istanbul, Turkey and an M.S. degree in Computer Science from Rensselaer Polytechnic Institute, Troy, NY. Dr. Kosar has received his Ph.D. degree in Computer Science from University of Wisconsin-Madison. He is the primary designer and developer of the Stork distributed data scheduling system which has been adopted by many national and international institutions. Dr. Kosar holds several prestigious awards for his work on data-aware distributed computing, including NSF CAREER Award, LSU Rainmaker Award, LSU Flagship Faculty Award, and LSU CCT Faculty of the Year Award.