University of Augsburg
Prof. Dr. Hans Ulrich Buhl
Research Center
Finance & Information Management
Department of Information Systems
Engineering & Financial Management

Discussion Paper WI-235

Using a Grid for Risk Management: Communication Complexity of Covariance Calculations

by

Gilbert Fridgen

Presented at: 15th Americas Conference on Information Systems, AMCIS, San Francisco, California, August 2009

Universität Augsburg, D-86135 Augsburg
Visitors: Universitätsstr. 12, 86159 Augsburg
Phone: +49 821 598-4801 (Fax: -4899)
www.fim-online.eu

Using a Grid for Risk Management: Communication Complexity of Covariance Calculations

Gilbert Fridgen
FIM Research Center Finance & Information Management,
Department of Information Systems Engineering & Financial Management,
University of Augsburg, Germany
gilbert.fridgen@wiwi.uni-augsburg.de

ABSTRACT

Risk management has evolved as one of the key success factors for enterprises especially in the financial services industry. It is highly demanding in terms of business requirements and technical resources, making it an almost ideal application for distributed computing concepts like e.g. grid computing. In this paper we focus on a specific problem—the estimation of covariance matrices that provide a powerful tool for decisions on investments. In this context we analyze different network topologies that the corresponding calculations can be performed on. We derive complexity classes for a distributed calculation scenario on these topologies. As a general result we find an upper and lower bound for the complexity of a distributed calculation in an arbitrary network structure. These results not only provide a different view on grid resource allocation but also make a contribution towards better understanding the business value of grid computing.

Keywords

Grid Computing, Risk Management, Parallel Algorithms

INTRODUCTION

Whereas in the beginning grid computing was restricted to large-scale scientific applications, it has over the past years evolved to an increasingly relevant technology for the commercial sector. It is to some extent difficult to differentiate grid computing from related concepts of e.g. distributed computing, cluster computing, utility computing, or Service-Oriented Architectures. The meaning we intend to convey by our use of the term grid computing in this article is best captured by a definition of Buyya, 2004: 'A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed, autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements'. An overview of the status quo and current applications of grid computing provide e.g. Berman et al., 2003, Foster and Kesselman, 2003, and Abbas, 2004.

The availability of grid enabled business applications is a critical success factor for the wider adoption and further development of this powerful technology. One of the most promising domains for grid computing is the financial services industry with its information-driven business models and high needs for computing power. In fact this sector is often mentioned among the key industries for grid computing applications (Ricadela, 2002; Smith and Konsynski, 2004; Friedman, 2003). In this context resource-intensive risk management applications seem to be especially suitable. With the potentially huge amount of computing capacity a grid infrastructure offers (embracing resources of the whole enterprise or even of external providers) they can possibly be accelerated dramatically. Yet, even on a grid infrastructure resources are not unlimited. Thus it is necessary to consider the complexity associated with the corresponding calculations. For our analysis we focus on a specific problem in risk management—the calculation of covariance matrices that, on the one hand, provide a powerful tool for decisions on investments but, on the other hand, are a very complex and time-consuming assignment.

Principles of Risk Management

Enterprises are investing capital into risky investment objects in order to generate cash inflows and subsequently to increase the return of the invested capital. Especially with the background of the financial crisis and following the argumentation of Wilson, 1996, pp. 194 it is therefore crucial for the survival and success of an enterprise to be able to allocate the available capital to the right combination of investment objects taking into account their specific contributions to the overall risk. In order to abide by given risk limits the knowledge of the current overall risk position is an essential prerequisite. Consider for example a trading unit that needs exact and timely information about its risk exposure when deciding on security or option trades.

Finance offers a set of instruments to calculate risk measures, some of them seeming to be promising with regard to grid computing concepts. In this paper we focus on the 'variance-covariance approach'. When measuring portfolio risk it is not sufficient to take into account the variances of the investment objects' periodical returns alone. Instead, also correlation effects that exist when returns are not perfectly positively correlated have to be considered which can be achieved using covariance matrices.

The Variance-Covariance Approach

We can represent risky investment objects by stochastic variables. Typically historical data are used to derive a distribution for a stochastic variable. It is then possible to determine the actual risk position of the investment objects or to extrapolate from history to the future for the support of investment decisions. For two stochastic variables, X_i and X_j, we define (with $E[X_i]$ and $E[X_j]$ denoting the expected values of the variables) the covariance as

$$Cov[X_i, X_j] := E\Big[\Big(X_i - E[X_i]\Big)\Big(X_j - E[X_j]\Big)\Big]$$

Writing Cov_{ij} instead of $Cov[X_i, X_j]$ and σ_i^2 short for the variance of X_i we can determine the overall risk of a portfolio, σ_P^2, consisting out of k investment objects (numbered from 1 to k) as

$$\sigma_P^2 = \sum_{i=1}^{k} \sigma_i^2 + \sum_{i=1}^{k} \sum_{j=1;j\neq i}^{k} \mathrm{Cov}_{ij} = \sum_{i=1}^{k} \sum_{j=1}^{k} \mathrm{Cov}_{ij}$$

The instrument of covariance matrices is not restricted to the area of security portfolios. Instead all investments of an enterprise, like credit decisions in banks or even customers or projects can be seen as components of the enterprise's overall investment portfolio, having a return and a variance.

Characteristics of the Calculation of a Covariance Matrix

The covariance of two investment objects is not previously known and additionally changing over time (heteroscedasticity). Therefore covariances have to be empirically estimated by analyzing historical data. Using the basic approach for estimating covariances, the covariance matrix can also be calculated by

$$Cov_{ij} = E[X_i \cdot X_j] - E[X_i] \cdot E[X_j].$$

While the expected values $E[X_i]$ only have to be calculated once for each investment object, the expected value $E[X_i \cdot X_j]$ has to be calculated for each pairwise combination of investment objects. This shows that while the calculation of a single covariance is not very resource intensive, the calculation of a whole covariance matrix is are more complex problem. Although the equality $Cov_{ij} = Cov_{ji}$ makes the matrix symmetric, for k investment objects the total number of covariance calculations is still given by $k(k+1)/2$.

MODELLING APPROACH

An algorithm for computing covariances in parallel must tackle questions like which grid resources to allocate to the calculation of certain parts of the covariance matrix and how to distribute the input data necessary to perform these calculations. Both questions are inseparably intertwined since calculating part of the matrix requires only part of the input data and unnecessary transportation of input data should be avoided. These tasks are usually done by a grid middleware. However, the proposed problem requires too much domain knowledge for a standard grid middleware to distribute it efficiently.

The grid infrastructure considered for covariance calculation consists of n computing nodes and is depicted by an undirected and connected graph. Its vertexes model the nodes while the edges represent physical connections between the nodes. Loops and multiple edges connecting two vertexes are not allowed in our model.

Advancing towards the complexity of grid-based risk management calculations we restrict our considerations on *communication complexity* i.e. the amount of data (we will concretize the unit for measuring the 'amount of data' in the following section) that has to be exchanged between the nodes of the grid. For a *complexity class* we use the common asymptotic notation with Ω, O or Θ denoting the lower, upper or tight bound for the complexity class respectively. It is not surprising that complexity varies for different grid network *topologies*. Therefore, we propose calculation algorithms for a set of standard topologies to find their complexity.

Preliminaries

From here on we will use the term *investment object* synonymously with the investment objects' historic quotations. We assume the number of investment objects, k, and the number of nodes in the grid network, n, to be 'sufficiently' large. When necessary to ease our calculations we consider all variables $\in \mathcal{R}$ and omit small constant addends as it is common practice in complexity theory. For simplicity and to avoid case differentiations without relevance we implicitly assume an odd or even number of nodes n as needed.

We refer to the investment objects for which one node holds the historic quotations as to the node's *package*. The number of packages then equals the number of nodes, n. We define a *message* as the transmission of one package over one edge and the *distance* between two nodes as the number of edges/messages to get from one node to the other. We can then measure complexity by the number of messages necessary to calculate one covariance matrix (this connection obviously only holds if all messages are of the same size or at least order of size which is assured for our model by assumption 3). $C^{[topology]}$ denotes the complexity for a specific topology while C denotes the complexity for an arbitrary topology. We use the terms 'complexity', 'amount of communication' and 'number of messages' synonymously.

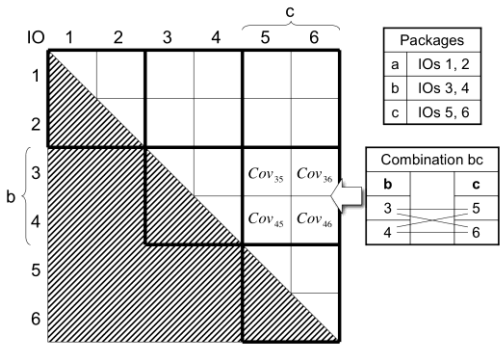

Figure 1. Covariance matrix of 6 investment objects (1-6) in 3 Packages (a-c)

The nodes holding a certain package are denoted by uppercase Latin letters $A, B, C, \ldots$, the packages with the corresponding lowercase Latin letters $a, b, c, \ldots$. We identify investment objects in our examples with numbers $1, 2, 3, \ldots$. Figure 1 shows an example of a covariance matrix.

On the package level we constitute a *package combination* out of two packages by calculating the pairwise covariances of each investment object in the one package with each in the other. For our complexity considerations it is then sufficient to analyze the package combinations rather than the covariances themselves. By building all package combinations we get one complete covariance matrix.

Assumption 1 *Covariances are calculated for a given and fixed number $k >> 1$ of investment objects. The necessary input data is available in the form of historic quotations.*

Assumption 2 *The grid infrastructure consists out of n nodes with $1 << n \leqq k$, each with the same constant computing capacity available for covariance calculations.*

Assumption 3 *The investment objects are distributed equally over the network, i.e. each node holds a package of the same size k/n.*

Assumption 4 *All nodes are used equally if every node calculates the same number of package combinations.*

This is obviously simplifying matters but without significant influence on the results. By concentrating on package combinations, we measure complexity depending on the number of nodes. Intuitively, it might be more interesting to consider the complexity depending on the number of investment objects k. We implicitly assume proportional growth of the number of nodes (e.g. workstations) and the number of investment objects, as both values somehow represent a company's size. Therefore, there exists a constant package size k/n which can be used to transform the complexities. The complexity class is not affected by this transformation.

Similar Problems and Other Publications

There are basically two areas that cover parts of the given problem: There is research on algorithms and routing on parallel infrastructures and there is research on grid computing covering especially questions of resource allocation.

In the relevant literature for grid networks one can find many articles addressing allocation problems for certain grid applications. For example Buyya et al., 2002 give an overview about existing systems and their underlying economic models. In particular auctions as described e.g. in Wolski et al., 2001 or agent technologies as proposed in Foster et al., 2004 are used to automatically allocate the available resources efficiently. However these publications try to find general ways to prioritize applications without analyzing a specific application domain whereas this paper analyzes a specific problem's complexity. Yu and Buyya, 2005 present a taxonomy of workflow management systems for grid computing. While neither of these systems specifically addresses our problem, they can be used for implementing our resulting algorithms.

Publications in the area of parallel algorithms cover parts of our problem in very specific ways. Algorithms for calculating convolutions could be adapted to calculate covariances as it is necessary for both problems to build pairwise combinations of the input data. Such algorithms have been intensively discussed, most times with the application of integer multiplication, e.g. in Atrubin, 1965 and Cappello and Steiglitz, 1983. The question of routing packets through a network of processors has already been analyzed e.g. by Valiant and Brebner, 1981 in a general way, by Krizanc et al., 1988 for mesh-connected arrays and by Leighton, 1990 for greedy algorithms on arrays. Leighton, 1992 gives an overview on more algorithms for parallel architectures. All of these algorithms were created for specifically designed or at least prior known topologies. A grid network is by nature inhomogenous, so these algorithms are not directly applicable. In contrast, our model identifies the complexity for communication in an arbitrary and possibly heterogeneous network infrastructure.

There are also several articles dealing with questions of data distribution in distributed systems in general and independent from a given problem. Basically two problems seem to be similar to a distributed calculation of a covariance matrix: The 'all-pairs shortest path' problem (APSP) and the 'optimal broadcast' problem. An APSP algorithm finds the shortest paths between every distinct pair of nodes in a graph (for recent research on APSP algorithms refer to Demetrescu and Italiano, 2004). This might come useful for our problem as all pairs of packages have to be brought together. Nevertheless, this perception disregards the fact that nodes can store and forward information. To distribute a package to several nodes it actually might be more efficient to accept a longer path but at the same time to reach more nodes on the way. According to assumption all nodes have to receive sufficient data to calculate the same number of combinations but an APSP algorithm does not give us any information where to calculate what combinations.

A broadcast algorithm distributes data available at a single node to all the other nodes efficiently. According to Awerbuch et al., 1990; Awerbuch and Schulman, 1997; Awerbuch et al., 1998 and Tiwari, 1987 the complexity for a single node distributing its data to all other nodes in an arbitrary network structure with n nodes is $\Theta(n)$. For the distributed calculation of a covariance matrix n nodes have to distribute their data to several others so a tight bound of $n \cdot \Theta(n) = \Theta(n^2)$ could be expected for our problem.

Buhl et al., 2009 quantify the utility of grid-based covariance calculations for risk management. As there are, to the best of our knowledge, no further articles covering the complexity of grid-based covariance calculation, we provide the first paper in this research area.

COMPLEXITY FOR DIFFERENT TOPOLOGIES

We will examine several idealized standard grid network topologies and propose different algorithms for distributed covariance calculations in order to analyze the complexity. It is already pointed out that we would expect upper and lower bounds of $O(n^2)$ and $\Omega(n^2)$, respectively, adapting the models of Awerbuch et al., 1990; Awerbuch and Schulman, 1997; Awerbuch et al., 1998 and Tiwari, 1987.

Conjecture 1 *The lower bound for the complexity class in the distributed computation scenario of a covariance matrix in an arbitrary network structure is* $\Omega(n^2)$*, the corresponding upper bound is* $O(n^2)$.

Fully Connected Topology

In the fully connected topology every node has direct connections to all other nodes. Every single package can be transmitted from one node to an arbitrary other by one message. Thus minimal communication is accomplished when sending every node exactly the packages for it to perform its calculations. We use this general idea to establish the theoretically minimal communication necessary.

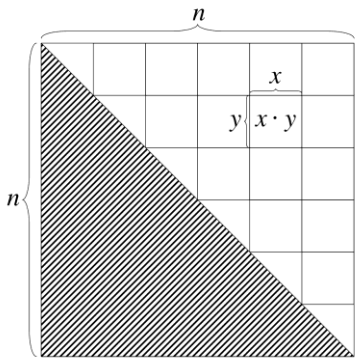

Figure 2. Schematic covariance matrix

We consider figure 2 that shows a schematic covariance matrix for n packages. As the matrix is symmetric only $n^2/2$ package combinations have to be calculated. Thus, according to assumption 4 each of the n nodes has to accomplish $n/2$ package combinations—in effect one half of the matrix is divided into n areas of equal size $n/2$. We will disregard the fact that every node already holds one package. Receiving $x + y$ distinct packages, a node can then calculate $x \cdot y$ package combinations. Thus, $x + y$ represents the number of packages necessary to accomplish $x \cdot y = n/2$ package combinations. It is immediately obvious that for a given and fixed $x \cdot y$ the term $x + y$ is minimized for $x = y$. Therefore, with $x \cdot y = n/2$ and $x = y$, we can state that the minimal number of messages c_{sq} for one node to calculate its required package combinations (a square out of the matrix) is determined by

$$x^2 = \frac{n}{2} \quad \Leftrightarrow \quad x = \sqrt{\frac{n}{2}} \quad \Rightarrow \quad c_{sq} = x + x = 2\sqrt{\frac{n}{2}} = \sqrt{2n} \tag{1}$$

Nodes computing package combinations at the triangles in the diagonal of the covariance matrix will need comparatively more messages to accomplish their (non-quadratic) share. We will neglect this minor inaccuracy—the additional communication for this purpose will only increase linearly with n—and determine the complexity for n nodes as

$$C^{full} \approx n \cdot c_{sq} = n \cdot \sqrt{2n} = \sqrt{2} \cdot n\sqrt{n} \tag{2}$$

As we were minimizing communication when designing this algorithm, we found the lower bound for a calculation on a fully connected topology to be $C^{full} \in \Omega(n\sqrt{n})$. As the algorithms are only sensitive to n, the upper bound equals the lower bound: $C^{full} \in O(n\sqrt{n})$. As all other topologies can be built from the fully connected topology by omitting edges, the fully connected topology must allow the minimal amount of communication possible. So, at the same time, we found the lower bounds for C in the general case: $C \in \Omega(n\sqrt{n})$.

This implicitly disproves conjecture 3.1, since for our problem an even lower complexity than $\Omega(n^2)$ can be reached. We will now analyze if (and if so how) this minimal complexity can be reached in other topologies.

Star Topology

The star topology is characterized by one 'hub' with direct connection to all other nodes. Hence, it has a direct neighborhood of $n - 1$ nodes. The neighbors themselves are not directly adjacent to each other. The minimal communication necessary in the star topology for calculating the covariance matrix is very similar to the communication considered in the fully connected topology. The hub can receive all packages with one message each. This results in a complexity of $c_1 = n - 1$. As soon as the packages are collected at the hub each package can be transmitted to any node with one message just as in the fully connected topology ($c_2 = C^{full}$). The hub itself already has all the data it needs: thus according to equation 1 we have $c_3 = -c_{sq}$.

$$\begin{aligned} C^{star} &= c_1 + c_2 + c_3 = n - 1 + n \cdot \sqrt{2n} - \sqrt{2n} \\ &= \sqrt{2} \cdot n\sqrt{n} + n - \sqrt{2n} - 1 \end{aligned} \tag{3}$$

The proposed algorithm for a star topology therefore realizes $O(n\sqrt{n})$ and thus also the theoretical lower bound.

Line Topology

In the line topology with n nodes $n-2$ nodes have a degree of 2 and two nodes have a degree of 1. The algorithm for distributed calculation in a line topology is more complicated than in the two previous topologies. Basically there are two reasons for sending a package to a node: The node needs a package to calculate package combinations or the node has to relay the package to a neighbor. We will call the two nodes at the end of the line (each with a degree of 1) the 'outmost' nodes (in the example of figure 3: nodes A and G). For them it makes no sense to forward any packages as the only nodes they could send packages to are the nodes they received the same packages from. Thus the only reason for sending data to the outmost nodes is their calculations to be done. On the other hand, the outmost nodes are not able to calculate all package combinations involving their own packages so they have to pass their packages on to their neighbors. As soon as this is finished one can treat the outmost nodes and their direct neighbors as a 'black box' (here: $A+B$, $F+G$). Again it makes no sense to send these black boxes more packages than they need to perform their calculations. And again they will pass their own data to their neighbors (C and E). As soon as this process is finished, one can again treat the black boxes and their direct neighbors like a larger black box (here: $A+B+C$, $E+F+G$). This can be iterated until the central node is reached from both directions, and then exchanges the data between the two halves of the line.

In order to determine the complexity of this algorithm we will disregard smaller inaccuracies, like there are more computations possible with the available data than with the available computing power or the fact that some combinations may already have been computed at on the other half. Quantifying the communication to the inside, denoted by c_1 is like sending all packages to a central node which has the position $(n+1)/2$ (from both ends of the line).

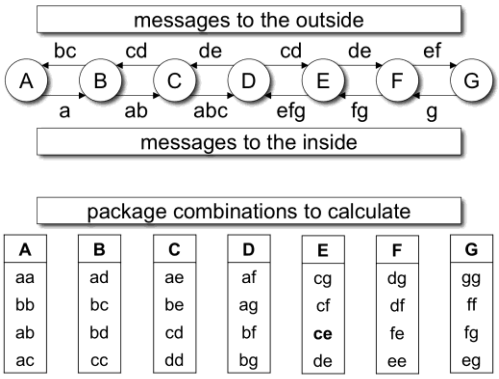

Figure 3. Communication in a line topology

In the line topology, the communication can be expressed as a sum over the distances between all nodes and the central node. On each side there are $(n-1)/2$ nodes. As the nodes are ordered in a line, when going from the calculating node to the outside of the line, each node's distance to the calculating node increases by one. The overall complexity can thus be described by twice the sum over an increasing control variable i:

$$c_1 = 2 \cdot \sum_{i=1}^{\frac{n-1}{2}} i = \frac{n^2-1}{4} \tag{4}$$

Quantifying the communication to the outside, denoted by c_2, is more complex. The first question arising is the number of packages to send into a black box. We define ψ as the iteration of our black box, with the first iteration ($\psi = 1$) describing a black box containing the outmost node, the second one describing a black box containing the outmost node and its direct neighbor and so on. We also define $n_b(\psi) = \psi$ as the number of nodes in a black box, $p_b(\psi)$ as the number of combinations to be calculated in the black box, $q_b(\psi)$ as the minimally necessary number of packages to do these calculations (compare to equation 1), and finally $c_b(\psi)$ as the necessary number of messages that have to be sent to the black box.

$$\begin{aligned} p_b(\psi) &= \frac{n+1}{2} \cdot n_b(\psi) = \frac{n+1}{2} \cdot \psi \\ q_b(\psi) &\approx \sqrt{2p_b(\psi)} = \sqrt{(n+1)\cdot\psi} \\ c_b(\psi) &= q(\psi) - n(\psi) = \sqrt{(n+1)\cdot\psi} - \psi \approx \sqrt{n\cdot\psi} - \psi \end{aligned}$$

The parameter c_2 can be quantified by summing up the $c_b(\psi)$ for all the black boxes. As mentioned above there are black boxes for $\psi = 1, \ldots, (n-1)/2$ and as the algorithm is applied from both sides of the line we have to count every black box twice.[1]

$$c_2 \approx 2 \cdot \sum_{\psi=1}^{\frac{n-1}{2}} \left(\sqrt{\psi \cdot n} - \psi \right) = 2 \cdot \left(\sum_{\psi=1}^{\frac{n-1}{2}} \sqrt{\psi \cdot n} - \sum_{\psi=1}^{\frac{n-1}{2}} \psi \right) \approx \frac{1}{3}\sqrt{2}n^2 - \frac{4}{3}\sqrt{n} - \frac{n^2-1}{4} \tag{5}$$

With c_1 and c_2 we can calculate the total number of messages for a line topology.

$$C^{line} = c_1 + c_2 = \frac{n^2-1}{4} + \frac{1}{3}\sqrt{2}n^2 - \frac{4}{3}\sqrt{n} - \frac{n^2-1}{4} = \frac{1}{3}\sqrt{2}n^2 - \frac{4}{3}\sqrt{n} \tag{6}$$

According to $C^{line} \in O(n^2)$ for our proposed algorithm we have the first topology not allowing a distributed calculation in $O(n\sqrt{n})$.

Ring Topology

The ring topology is a ring of n nodes with a degree of 2 each. The ring topology is quite similar to the line topology as it only consists out of one more edge, connecting the two ends of the line. As we can easily transform a ring into a line, the complexity for a ring topology is surely not higher than the complexity for a line topology.

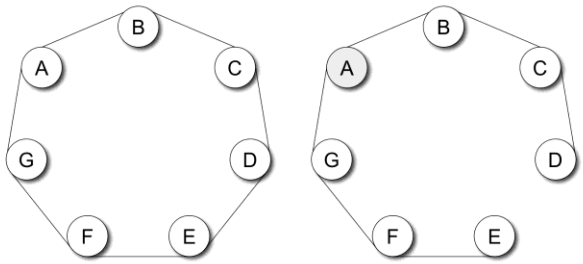

Figure 4. Example of a ring topology vs. line topology

Having a closer look on the ring topology we can state that the maximal distance between two nodes is $(n-1)/2$. This means that either A's package has to be transported to D, or vice versa or A's and D's packages have both to be transported to B or to C. Either way this causes $(n-1)/2$ messages and for n nodes $n(n-1)/4$ messages. One can easily see, if every node behaves equally, every package combination can be calculated and all nodes and edges are equally utilized. The total number of messages for a ring topology can thus be described by $C^{ring} = n(n-1)/2$ which is $O(n^2)$, too.

Tree Topology

We define the tree topology as a network structure whose nodes are arranged in a perfect λ-ary tree with a depth δ. The proposed algorithm is very similar to the line topology's. Its calculation first requires to determine the dependencies of the number of nodes n, the depth of the tree δ and the order of the tree $\lambda > 1$.

$$n = \sum_{i=0}^{\delta} (\lambda^i) = \frac{\lambda^{\delta+1}}{\lambda - 1} \tag{7}$$

By solving equation 7 to δ we can determine the depth of a tree depending on its number of nodes and its order.

$$\delta = \log_\lambda(n \cdot (\lambda - 1) + 1) - 1 \approx \log_\lambda(n\lambda) - 1 = \log_\lambda(n) \cdot \log_\lambda(\lambda) - 1 \approx \log_\lambda(n) \tag{8}$$

The number of messages to the root, c_1, is then given by the tree's depth sum (the sum of all nodes' depths) and thus can be expressed as

[1]We approximate the sum by an integral with the same bounds.

$$c_1 = \sum_{i=1}^{\delta} i \cdot \lambda^i = \sum_{j=1}^{\delta} \sum_{i=1}^{\delta} \lambda^i = ... = \frac{\delta \cdot \lambda^\delta \cdot \lambda}{\lambda - 1} - \frac{\lambda^\delta \cdot \lambda - \lambda}{(\lambda - 1)^2} \approx n \cdot \log_\lambda(n) - \frac{n-1}{\lambda - 1} \tag{9}$$

The amount of communication to the other nodes is established by the least number of packages necessary to do all computations there. In order to determine the number of packages, we first need to quantify the number of nodes n_s, in a subtree with its root in depth φ. We have $n_s(\varphi) = 1/\lambda^\varphi \sum_{i=\varphi}^{\delta} \lambda^i = \sum_{i=0}^{\delta-\varphi} \lambda^i$. We define, in analogy to the line topology, $p_s(\varphi)$ as the number of calculations to be done in the subtree, $q_s(\varphi)$ as the necessary number of packages for these calculations and $c_s(\varphi)$ as the necessary packages to transfer into the subtree.

$$\begin{aligned} p_s(\varphi) &= \frac{n+1}{2} \cdot \sum_{i=0}^{\delta-\varphi} \lambda^i \\ q_s(\varphi) &= \sqrt{2p_s(\varphi)} = \sqrt{(n+1) \cdot \sum_{i=0}^{\delta-\varphi} \lambda^i} \\ c_s(\varphi) &= q_s(\varphi) - n_s(\varphi) = \sqrt{(n+1) \cdot \sum_{i=0}^{\delta-\varphi} \lambda^i} - \sum_{i=0}^{\delta-\varphi} \lambda^i \end{aligned} \tag{10}$$

The amount of communication to the outside c_2, is the sum over all c_s for all depths and all subtrees at a specific depth.

$$c_2 = \sum_{\varphi=1}^{\delta-1} \left(\lambda^\varphi \cdot \left(\sqrt{(n+1) \cdot \sum_{i=0}^{\delta-\varphi} \lambda^i} - \sum_{i=0}^{\delta-\varphi} \lambda^i \right) \right) \approx n^{\frac{3}{2}} \cdot \frac{\sqrt{\lambda}}{\sqrt{\lambda} - 1} - n \cdot \log_\lambda(n) + \frac{n-1}{\lambda - 1} - n \cdot \frac{\sqrt{\lambda}}{\sqrt{\lambda} - 1} \tag{11}$$

The overall number of messages necessary to perform a distributed calculation in a perfect tree is constituted as

$$C^{tree} = c_1 + c_2 \approx n^{\frac{3}{2}} \cdot \frac{\sqrt{\lambda}}{\sqrt{\lambda} - 1} - n \cdot \frac{\sqrt{\lambda}}{\sqrt{\lambda} - 1} \tag{12}$$

Surprisingly, the perfect tree structure can also realize $O(n\sqrt{n})$ although there is no direct similarity to the algorithms for the fully connected or the star topology.

GENERIC COMPLEXITY CONSIDERATIONS

Topology	Fully Connected	Star Topology	Line Topology	Ring Topology	Tree Topology
Complexity	$O(n\sqrt{n})$	$O(n\sqrt{n})$	$O(n^2)$	$O(n^2)$	$O(n\sqrt{n})$

Table 1. Communication Complexities of the Analyzed Topologies

Assuming optimality for the algorithms proposed in section 3, there is some interesting results worth to be pointed out. There seems to be no clear coherence between complexity and the number of edges relative to the number of nodes. In this section we will try to find a general way to determine the upper and lower bounds of complexity for an arbitrary topology. For enabling our proof we need an assumption regarding the availability of data.

Assumption 5 *Each node receives the packages it needs for its calculations from the nodes with the smallest distances first. Each of these nodes holds at least one required package.*

Obtaining packages from the nodes within the smallest distances first is certainly a good idea for minimizing the complexity. All required packages are available for the first node receiving packages as any package is usable for calculation (no combinations have been calculated yet) and every node holds exactly one package. Taking this finding as a starting point we can make

Proposition 1 *The complexity for the distributed calculation of a covariance matrix in an arbitrary network structure is* $\Omega(n\sqrt{n})$ *and* $O(n^2)$.

We will prove these upper and lower bounds using the following

Lemma 2 *The marginal number of messages necessary for a single node in an arbitrary topology that receives i packages is at least constant and grows at most linearly with each additional i.*

Proof 3 *We define the distance $d(A, Z)$ between two (not necessarily distinct) nodes A and Z. The number of messages necessary for A to receive a package from Z equals A's distance to Z. To minimize communication A will receive the packages of the nodes with the smallest distances first. As A is part of a connected graph it has at least one direct neighbor B with $d(A, B) = 1$. In the best case each package is available within a distance of 1 and can thus be received with a constant number of messages (exactly 1 message). In the worst case every node holds only one usable package and the next reachable node C is not a neighbor of A but of B, so $d(A, C) = d(A, B) + d(B, C) = d(A, B) + 1 = 2$. Again, in the worst case, the next reachable node D is neither neighbor of A or B but a neighbor of C a.s.o. Generally: In the worst case with N_i denoting the ith reachable node whose package is to be received at A: $d(A, N_i) = d(A, N_{i-1}) + 1 = d(A, N_{i-2}) + 2 = ... = d(A, N_1) + i - 1 = i$. As $d(A, N_1)$ is constant the necessary number of messages to receive i packages increases in the worst case linearly with each additional i.*

With lemma 4.2 and assumption 5 we can prove proposition 4.1.

Proof 4 *The lower bound of a distributed calculation $\Omega(C)$ has already been deduced before as $C = \sqrt{2} \cdot n\sqrt{n} \Rightarrow C \in \Omega(n\sqrt{n})$. We can derive the upper bound of a distributed calculation $O(C)$ using almost the same model as for the lower bound. Equation 1 delivers the necessary number of packages for calculating one node's workload (with $n/2$ package combinations) as $\sqrt{2n}$. According to lemma 4.2 the number of messages required to receive these packages grows at most linearly. Accounting for linear dependency in the worst case we derive an upper bound for the resulting complexity as*

$$C = n \cdot \sum_{i=1}^{\sqrt{2n}} i = n \cdot \frac{\sqrt{2n}(\sqrt{2n} + 1)}{2} \approx n \cdot \frac{\sqrt{2n} \cdot \sqrt{2n}}{2} = n^2 \tag{13}$$

As this is a worst case scenario we can state that $C \in O(n^2)$.

CONCLUSION

In this paper we restricted our analysis to one well-defined problem: the grid-based calculation of covariance matrices. Although covariances are widely used in financial applications, we thereby covered only a small subset of risk management methods and algorithms. Other approaches and applications for grid computing (like e.g. Monte-Carlo simulations which have a high parallelization potential as well) still have to be evaluated.

We considered different grid network topologies and scrutinized suitable algorithms for covariance calculation on these network structures. From there we derived the corresponding complexity classes for a distributed calculation on each topology. The basic algorithms proposed in this paper assume a very regular topology and an omniscient coordinator. Therefore none of them will be directly applicable to a real company's complex infrastructure. Nevertheless, when perceiving the nodes of the topology not as single workstations but as whole subsidiaries of a large enterprise an e.g. line topology is absolutely realistic. Furthermore the resulting complexity classes could in combination with a quantification for the benefits of the calculation be utilized for the determination of an optimal investment in covariance calculations.

For an arbitrary topology these results are as well meaningful: For example when designing a specialized algorithm for a company's network infrastructure one can apply the insights generated by the algorithms proposed (e.g. that it makes sense to send as few packages to the outmost nodes as possible). Furthermore, even if in a company's network each node's effort to receive data grows in a logarithmic fashion (e.g. if each node's degree is greater than 2) one can use $O(n\sqrt{n})$ as a benchmark for the design of a specific covariance calculation method.

ACKNOWLEDGEMENTS

A prior version of this research has been presented at the 7th International Conference on Optimization: Techniques and Applications (ICOTA7), Kobe, (Japan), December 2007. Thank you to the audience and especially to Dr. Hackenbroch for very useful feedback.

REFERENCES

1. Abbas, A. (2004) Grid Computing: A Practical Guide to Technology and Applications, Charles River Media, Hingham.

2. Atrubin, A. (1965) A one-dimensional real-time iterative multiplier, *IEEE Transactions on Electronic Computers* (EC-14:3), 394–399.
3. Awerbuch, B., Cidon, I. and Kutten, S. (1990) Optimal Maintenance of Replicated Information, in *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, Comp. Soc. of the IEEE, IEEE, 492–502.
4. Awerbuch, B., Cidon, I., Kutten, S., Mansour, Y. and Peleg, D. (1998) Optimal Broadcast with Partial Knowledge, *SIAM Journal on Computing* (28:2), 511–524.
5. Awerbuch, B. and Schulman, L. J. (1997) The Maintenance of Common Data in a Distributed System, *Journal of the ACM* (44:1), 88–103.
6. Berman, F., Fox, G. C., and Hey, A. J. G. (2003) Grid Computing; making the global infrastructure a reality, John Wiley & Sons Ltd., West Sussex.
7. Buhl, H. U., Fridgen, G., Hackenbroch, W. (2009) An Economic Analysis of Service-Oriented Infrastructures for Risk/Return Management, to be published in: *Proceedings of the 17th European Conference on Information Systems*, Verona, Italy, 2009.
8. Buyya, R. (2004) Grid Computing Info Centre: Frequently Asked Questions, http://gridcomputing.com/gridfaq.html.
9. Buyya, R., Abramson, D., Giddy, J. and Stockinger, H. (2002) Economic Models for Resource Management and Scheduling in Grid Computing, *The Journal of Concurrency and Computation: Practice and Experience* (14:Special Issue on Grid Computing Environments), 1507–1542.
10. Cappello, P. R. and Steiglitz, K. (1983) A VLSI layout for a pipelined Dadda multiplier, *ACM Trans. Comput. Syst.* (1:2), 157–174.
11. Demetrescu, C. and Italiano, G. F. (2004) A New Approach to Dynamic All Pairs Shortest Paths, *Journal of the ACM* (51:6), 968–992.
12. Foster, I., Jennings, N. R. and Kesselman, C. (2004) Brain Meets Brawn: Why Grid and Agents Need Each Other, in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, IEEE Computer Society, Washington, DC, USA, 8–15.
13. Foster, I. and Kesselman, C. (2003) The Grid 2: Blueprint for a new Computing Infrastructure, Morgan Kaufmann Publishers, San Francisco.
14. Friedman, M. (2003) Grid Computing: accelerating the search for revenue and profit for Financial Markets, *Building an Edge*, 143–145.
15. Krizanc, D., Rajasekaran, S. and Tsantilas, T. (1988) Optimal routing algorithms for mesh-connected processor arrays, in Reif, J. (Ed.), *Proceedings, 3rd Aegean Workshop on Computing: VLSI Algorithms and Architectures*, Springer-Verlag, Berlin, volume 319 of *Lecture Notes in Computer Science*, 411–422.
16. Leighton, F. T. (1990) Average case analysis of greedy routing algorithms on arrays, in *SPAA '90: Proceedings of the second annual ACM symposium on Parallel algorithms and architectures*, ACM Press, New York, NY, USA, 2–10.
17. Leighton, F. T. (1992) Introduction to parallel algorithms and architectures: array, trees, hypercubes, Morgan Kaufmann Publishers, San Mateo, CA, USA.
18. Ricadela, A. (2002) Living on the Grid, *InformationWeek* (893), 30–35.
19. Smith, H. and Konsynski, B. (2004) Grid Computing, *MIT Sloan Management Review*, Fall 2004, 7–9.
20. Tiwari, P. (1987) Lower Bounds on Communication Complexity in Distributed Computer Networks, *Journal of the ACM* (34:4), 921–938.
21. Valiant, L. G. and Brebner, G. J. (1981) Universal schemes for parallel communication, in *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, ACM Press, New York, NY, USA, 263–277.
22. Wilson, T. C. (1996) Calculating Risk Capital, in Alexander, C. (Ed.), *The Handbook of Risk Management and Analysis*, John Wiley & Sons Ltd., West Sussex, 193–232.
23. Wolski, R., Plank, J. S., Brevik, J. and Bryan, T. (2001) Analyzing Market-Based Resource Allocation Strategies for the Computational Grid, *International Journal of High Performance Computing Applications* (15:3), 258–281.
24. Yu, J., Buyya, R. (2005) A Taxonomy of Workflow Management Systems for Grid Computing, *Journal of Grid Computing* (3:3-4), 171-200.