Computational Risk Management for Building Highly Reliable Network Services

Brent N. Chun
Intel Research Berkeley
2150 Shattuck Ave. Suite 1300
Berkeley, CA 94704
bnc@intel-research.net

Philip Buonadonna
Intel Research Berkeley
2150 Shattuck Ave. Suite 1300
Berkeley, CA 94704
pbuonado@intel-research.net

Chaki Ng
Harvard University
Maxwell Dworkin 121
33 Oxford Street
Cambridge, MA 02138
chaki@eecs.harvard.edu

Abstract

Building reliable network services that can deliver consistent high performance to clients in the presence of failures and bursty demand is expensive and inefficient. Resources often need to be heavily overprovisioned to accommodate peak demand and the cost of such over-provisioning "prices out" many applications that could stand to benefit from a performance safety-net and ultimately provide more reliable service to end users. To address these problems, we propose an approach based on a shared Computational Service Provider (CSP). A CSP is an entity which provides massive amounts of widely distributed computation and storage and makes resources available through a mix of spot and derivative markets. Services obtain resources through the CSP and, drawing inspiration from finance, employ quantitative risk management techniques for trading off cost, performance, and risk to probabilistically achieve target levels of delivered client performance.

1 Introduction

Building reliable network services that can deliver consistent high performance to clients in the presence of failures and bursty demand is expensive and inefficient. Resources often need to be heavily overprovisioned to accommodate peak demand and the cost of such over-provisioning "prices out" many applications that could stand to benefit from a performance safety-net and ultimately provide more reliable service to end users. While shared infrastructures based on Service-Level Agreements (SLAs) are occasionally workable and leverage economies of scale, SLAs are generally static, long-term contracts. Consequently, they are typically inefficient (due to overprovisioning in the contracts) and/or ineffective at responding to time-varying load in a predictable manner.

To address these problems, we propose an approach based on a shared Computational Service Provider (CSP). A CSP is an entity which provides massive amounts of widely distributed computation and storage and makes resources available through a mix of spot and derivative markets [1]. Services obtain resources through the CSP and, drawing inspiration from finance, employ quantitative risk management techniques for trading off cost, performance, and risk to probabilistically achieve target levels of delivered client performance. The end goal is to reap the benefits of statistical multiplexing in the CSP while also giving applications the infrastructure and tools to quantitatively manage their resource provisioning in the face of time-varying resource supply and demand.

The rest of this paper is organized as follows. In Section 2, we draw analogies to finance, describe the two key metrics we use to quantify performance risk, and describe the resulting optimization problem that services face over time. In Section 3, we then describe the infrastructure needed to support implementation of

[1] A spot market is a market in which commodities (e.g., computational resources) are sold and delivered immediately. In contrast, an example of a derivatives market is a futures market which trades futures contracts, each of which is an agreement to buy or sell an asset at certain time *in the future* at a certain price.

the aforementioned risk management strategies. Specifically, this includes the types of markets supported by a CSP, the supply and demand forecasting algorithms that will be needed, and touches on programming risk management strategies to probabilistically achieve target levels of client performance. Lastly, in Section 4, we conclude and discuss open problems that need to solved in order to make this approach not only tenable but effective in practice.

2 Computational Risk Management

While delivering consistent high performance to clients is a desirable property for a reliable network service to have, a key issue faced by a service is the cost associated with delivering this performance. Depending on the service and the context, the budget available for acquiring and managing resources to deliver target levels of performance vary, often by wide margins. For example, consider the cost/performance trade-off of ensuring that 99.9% of client requests get processed within 10 ms in an electronic equities trading system versus a small, local e-commerce provider. In this case, the cost associated with not processing requests in a timely manner are likely to be significantly higher for the trading system. Consequently, such a system is likely to require a larger budget for provisioning and managing resources such that good performance is ensured with high probability.

2.1 Performance Value-at-Risk (VaR)

Formally, we can express this inherent trade-off between cost and performance as follows: given a desired target level of performance V, a time horizon N, and probability X of meeting the performance target, how should resources be provisioned over time such that we can say that we are $X\%$ certain that we will deliver at least performance V over the next N time units? For example, in the trading system, we may wish to say: "We are 99.9% certain that we will process all trades in under 10 ms over the next month". One way to achieve this is to provision the system with infinite resources, an option that is neither feasible nor desirable given its high cost and our finite service budget B. Hence, in reality both the resources available and the budget for acquiring those resources will be finite. Since client demand can be bursty, the necessary resources at any given time to meet the desired performance requirements also varies. Thus, we have a resource allocation problem, one which involves uncertainty in both resource demand and supply over time.

In finance, money managers face an analogous problem. There, given a portfolio of financial instruments (e.g., stocks, bonds, derivatives, etc.), they wish to quantitatively assess the risk associated with the portfolio. The Value-at-Risk (VaR) [7] of a portfolio summarizes its total risk based on the statement: "We are $X\%$ certain that we will not lose more than V dollars in the next N days" (e.g., $X = 99\%$ and $N = 10$ days are typical numbers). The goal in managing the portfolio is then to make appropriate trade-offs between X, V, and N through intelligent composition and management of the portfolio. Typically, this is done by constructing appropriate mathematical models and subsequently hedging risk using derivatives, purchasing specific quantities of stocks that are negatively correlated in price, and so forth. The fundamental mechanism is trading and the infrastructure that enables it is markets for a broad range of financial instruments.

Given the similarities above, we thus propose the notion of a *performance VaR* which characterizes target levels of performance as described in our trading example above. Given a target performance VaR, our goal then is to provide the necessary infrastructure so resources for the service can be automatically provisioned such that the VaR is met (Section 3). While this paper focuses exclusively on the performance VaR, in principle our approach can be applied to other target metrics as well (e.g., availability, data reliability, etc.). Optimizing for combinations of metrics simultaneously is also a possibility but will require being able to model and decompose the various sources of underlying uncertainty and to understand their relative contributions to each component of a target multidimensional VaR.

2.2 Performance Conditional VaR (cVaR)

While the performance VaR tells us that $X\%$ of time we do not drop below some target level of performance V, it does not say anything about the range of performance delivered for the other $(100 - X)\%$ of client requests. Depending on the service, not meeting our performance goal by a small amount may be much more acceptable than missing our performance goal by a wide margin (e.g., dropping all $(100 - X)\%$ of all client requests in a trading system versus adding an additional 1 ms of processing per request).

Thus, in addition to the VaR, we would like a metric that characterizes our performance goals for these outlier cases. Since these cases are the uncommon case, this process can essentially be viewed as damage control where what we are trying to develop is a quantitative way of managing this damage control.

Not surprisingly, an analogous concept arises in finance. The Conditional Value-at-Risk (cVaR) [15] of a portfolio quantifies the expected loss on the portfolio conditioned on the loss exceeding V dollars. Applied in a computational setting, we thus propose the analogous idea of a *performance cVaR* which is the expected performance delivered over time period N averaged over all requests whose performance was worse than V.

2.3 Risk Management

The risk management problem is then twofold. First, given a target VaR and budget B for a particular network service, our first goal is to provision a distributed set of resources for the service such that VaR is satisfied in every time step over the time interval N while minimizing cost and meeting the budget constraint. We make no assumptions about the time delays to provision resources relative to N. Instead, we assume that the time to perform resource provisioning is factored into the portfolio of resource contracts obtained from the CSP such that "deltas" to get back to the target resource set can be made in a timely manner in response to observed failures while still maintaining the target VaR. Second, given the performance VaR, our second objective is to minimize the performance cVaR. Roughly, these goals translate to: (i) trying to deliver a consistent target level of performance to the majority of clients over some time period and (ii) trying to ensure that when we do not meet our performance targets that the resulting performance is not exceedingly bad.

To achieve these goals, we need access to a large pool of computational resources. This is required so we can accommodate a service's peak demands while also providing enough extra capacity such that failures and recovery can be handled while still delivering consistent high performance to clients. One way to accomplish this is to simply overprovision resources to handle peak demand. While such an approach is workable for certain critical network services where the costs are acceptable, it is still generally expensive to do, "prices out" many network services, and often requires significant internal expertise to build and manage such an infrastructure.

Consequently, we propose an alternative architecture based on Computational Service Providers (CSPs). In contrast to previous work [4], which proposed similar infrastructures, here we propose CSPs where resources are priced in both space and time and acquired through a set of markets which trade a mix of *computational instruments* (varying types of resource contracts). Through programmatic trading in such markets, online monitoring and performance forecasting, and optional service-specific information, our task is then to construct optimal portfolios of computational instruments that aim to meet our target performance VaR and cVaR.

3 Computational Service Providers

In the proposed model, network services use resources in the CSP to provide service to clients and services use quantitative risk management techniques while leveraging statistical multiplexing in the CSP to deliver target levels of performance VaR and cVaR at a minimum cost. Whereas in finance, the mechanism is trading and the infrastructure is equity and fixed income markets to manage financial risk, here we also use trading and manage computational risk using markets in computational resources. Using such markets, the idea is that service operators, or more likely programs or scripts acting on their behalf, construct and manage portfolios of computational instruments to achieve target levels of performance VaR and cVaR. Effective management of such portfolios will depend fundamentally on our ability to characterize resource supply and demand over time, in particular correlated failures within the CSP.

3.1 Statistical Multiplexing

The primary reason for the existence of a CSP is statistical multiplexing. It is more efficient to have multiple network services share a common infrastructure that can absorb failures and bursts in client demand than it is to have every service overprovision resources to accommodate peak resource requirements using internal infrastructure. The implication is that a CSP will typically aim to provide a pool of aggregate resources smaller than the sum of peak resource demands across all services. It will then rely on statistical multiplexing and revenue from clients and observed demand over time in deciding whether to extend/collapse resource capacity.

The user thus faces two key consequences: (i) the CSP cannot handle every possible mix of resource requests (e.g., all services requesting peak resource requirements to meet their target VaR and cVaR levels) and (ii) given that markets are the means by which resources are acquired, the cost of obtaining resources is a time-varying quantity that is a function of supply and demand. The implication is thus that users will need to perform computational risk management to achieve target performance VaR and cVaR levels as described in Section 2.3 while also being cognizant of finite resource capacity and time-varying supply and demand. This motivates the need for markets that enable flexible hedging (e.g., locking in 64 machines for two days in the future for some price in anticipation of a burst in client demand such as a new product release).

3.2 Computational Markets

Markets in computational instruments provide the infrastructure that allow users to manage target performance VaR and cVaR levels via trading of computational instruments. Example computational instruments might include contracts that provide dedicated use of a machine or persistent storage over specific periods of time. To allow services to dynamically acquire such resources (e.g., in response to a sudden burst in demand, or perhaps a trend indicating a general increase in demand), a CSP will minimally provide spots markets in node and storage hours. For example, a baseline contract in the node spot market might specify dedicated use of a particular node over time period $[t_1, t_2]$. We propose using standard contracts, despite potential fragmentation issues, to create market liquidity and promote efficiency.

In addition to spot markets, a CSP will offer at least a futures market and perhaps an options market. Futures markets trade futures contracts, each of which is an agreement to buy or sell an asset at certain time in the future at a certain price. In the case of a CSP, such a contract would specify use of a particular node over future time period $[t_1, t_2]$ at price p_f. These contracts allow users to hedge risk due to volatility in client demand for their service as well as aggregate client demand for shared resources which can cause adverse price movements and resource availability. As an example, suppose that a service operator knows that demand for her service is going to increase in 30 days due to a product launch. Rather than waiting until that time to purchase the resources, at which time the prices could have swung to unfavorably high levels due to external factors (e.g. other services release at around the same time, holiday seasons, etc.), she can instead purchase a set of futures contracts and lock in a fixed price. The VaR and cVaR of this new portfolio is hence improved for the period leading up to the launch due to the hedge. To understand the case when the actual price of the resources on launch is lower than what the user paid via the futures contract, one can draw an analogy to insurance, which protects the downside risk (what we really worry about), despite the fact that premiums paid may not be collectible should nothing happen.

Both spot and derivative markets will be two-sided so that both buyers and sellers can trade CSP resources, thereby enabling flexible trading scenarios to occur. In addition, services can resell excess resources should their needs change. Since the CSP owns the resources, it will control the supply of new computational instruments into the markets over time. It is also worth pointing out that the markets we propose consist of contracts for single-node resources. That is, we do not consider bids for baskets of resources (i.e., combinatorial bids) at the lowest level of the system. This is done primarily as an application of the end-to-end principle [16] and our belief that many production network services will derive incremental utility from partial allocations. For example, a user ideally might want 64 nodes and be willing to pay \$100 but may also be willing to accept 32 nodes for \$50 or any linearly scaled (#nodes,price) pair in between. Note that this stands in contrast to distributed computing environments such as network testbeds [5, 14, 18], where users frequently do seek to run on specific combinations of resources to perform specific experiments (e.g., scalability tests, tests using specific network topologies, etc.).

Trading for baskets of resources at the lowest level in a CSP does not preclude the use of computational risk management. Depending on the trading mechanism employed, it could make computing the cost to achieve some target level of VaR more complex however. For example, if the market supports trading of arbitrary baskets of resources, this makes quantifying the market value of an arbitrary basket of resources more difficult. In contrast, a baseline market that trades contracts for single-node resources is easier to reason about. With standardized contracts, historical prices can be used to gauge market demand for node resources and these prices can subsequently be used as part of portfolio optimization. Should users require complex resource combinations (e.g., "8 nodes for a day or none at all"), we believe that such requests can be accommodated through secondary markets operated by third parties using specialized market mechanisms such combinatorial auctions [6, 8, 10–12] or exchanges [13].

3.3 Forecasting Resource Supply and Demand

As part of managing VaR and cVaR, another key part of computational risk management is active monitoring and forecasting of both individual network service demand and aggregate resource demand over all services. The reason this is important is because VaR and cVaR are target metrics that are independent of workload size. They simply specify statistical requirements that must be met over any workload over a specified time period. A batch of resources that meets performance VaR and cVaR for an average workload may be completely insufficient for another workload corresponding to, say, a flash crowd. Consequently, absent external a priori knowledge about service demand, a key mechanism for managing risk will be active monitoring and forecasting of resource demand and using demand forecasts as part of active computational portfolio management.

Similarly, while the aggregate resource supply of a CSP is likely to be more stable than demand, it too is a time-varying quantity. Nodes in the system can crash, performance failures can occur, and both may be correlated across certain sets of machines at different points in time. Thus, while forecasting demand is probably more important, forecasting the resource supply offered by a CSP is also potentially relevant.

Forecasting demand and supply are not entirely new problems. Previous work, for example, has investigated profiling the resource requirements of network services [1, 3, 17] and characterizing the failure characteristics of large distributed systems [19]. The former can be viewed as providing individual data points for service demand given specific workloads, while the latter can be viewed as providing nearly the quantities we seek. In the case of forecasting resource demand, one possible solution for a CSP might be for the CSP to maintain and export resource usage statistics to applications, which might then use this information in combination with internal application-specific statistics (e.g., number of requests received over various time periods) to characterize their resource demands as a function of workload. In the case of forecasting resource supply, we could then leverage the results from [19]. Among other things, two key results of that paper were that an individual machine's MTTF and MTTR can be predicted with reasonable accuracy and that large-scale correlated failures are common in real systems. They tested these hypotheses on three widely distributed systems (PlanetLab [14], DNS, and the Web) and thus these results are likely to be widely applicable. The implication is that forecasting of resource supply in the aggregate is both practical and potentially important for managing performance VaR and cVaR in network services.

3.4 Programming

Finally, computational risk management is doomed to failure if it makes building reliable network services overly cumbersome and complex. While trading and computational markets by the CSP provides the mechanism and infrastructure to enable implementation of computational risk management strategies, ultimately the execution of those strategies needs to be programmed and executed to dynamically provision a service's resources over time. Specifically, this will entail: (i) forecasting resource demand and supply both for the service and the system in the aggregate (the latter may be provided as a shared service potentially), (ii) determining the appropriate incremental changes in a service's portfolio of computational instruments to meet VaR at minimal cost and within the service's budget constraint and to minimize cVaR, and (iii) performing appropriate trades in the CSP's markets to realize the changes computed in the second step. A key challenge will thus be designing protocols and APIs to assist in automating these steps as much as possible and composing the results.

While sophisticated users may use the CSP's protocols and APIs directly to implement risk management strategies, many users will be better served by third parties that specialize in portfolio management services. Such third parties are analogous to financial advisors with users providing high-level risk management objectives (e.g., the service's target VaR and cVaR) and the third parties implementing risk management strategies that meet these objectives on their behalf. Since it is unlikely that each application will require its own unique risk management strategy, portfolio managers may potentially leverage economies of scale by identifying risk management "design patterns" which lead to parameterizable risk management strategy implementations. Users would then have the option of picking and choosing the risk management strategy that is most attractive to them without being concerned with the details of how the strategy gets implemented through the markets on the CSP.

4 Conclusion and Open Problems

In this paper, we proposed an approach for building highly reliable network services that leverages a shared pool of computational resources accessible through markets and managed using quantitative risk management techniques that trade off cost, performance, and risk to probabilistically achieve target levels of performance. Our approach draws inspiration from analogous risk management techniques used in finance to quantitatively manage the risk associated with uncertainties in equity and fixed income markets. While we believe there are distinct merits to our approach, there are nevertheless more questions than answers at this point and further research exploring these issues is needed to assess its feasibility for real systems.

One set of questions relate to pricing: how should a CSP price resources in its spot and derivative markets, can we leverage analogous techniques in finance to assist in this process (e.g., Black-Scholes-Merton [2, 9] for pricing derivatives), and what will the dynamics of such prices look like in practice? Another class of questions concerns portfolio management including: what types of strategies will be effective and what will they entail, what type of resource forecasting algorithms are needed, how accurate do such algorithms need to be to be effective, and can we leverage strategies in finance for managing VaR and cVaR (or related metrics) in managing computational portfolios? Lastly, programming network services in this style will be a key challenge and could make or break the feasibility of the proposed approach. Two important questions here are thus: (i) what should the programming model, protocols, and APIs for a CSP and its markets look like and (ii) can we construct parameterizable classes of risk management strategies that make it easy for important classes of network services to use a CSP and deliver reliable, consistent performance to clients?

References

[1] M. Aron, S. Iyer, and P. Druschel. A Resource Management Framework for Predictable Quality of Service in Web Servers, 2000.

[2] F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81:637–659, May–June 1973.

[3] L. Cherkasova, W. Tang, and S. Singhal. An SLA-Oriented Capacity Planning Tool for Streaming Media

Services. In *Proceedings of the International Conference on Dependable Systems and Networks, (DSN-2004)*, June 2004.

[4] B. Chun, Y. Fu, and A. Vahdat. Bootstrapping a Distributed Computational Economy with Peer-to-Peer Bartering. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[5] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, May 2005.

[6] S. de Vries and R. V. Vohra. Combinatorial Auctions: A Survey. *INFORMS Journal on Computing*, 15:284–309, 2003.

[7] G. P. Hopper. Value at Risk: A New Methodology for Measuring Portfolio Risk. *Business Review, Federal Bank of Philadelphia*, July/August 1996.

[8] A. Kothari, D. C. Parkes, and S. Suri. Approximately-strategyproof and tractable multi-unit auctions. *Decision Support Systems*, 2004.

[9] R. C. Merton. Theory of Rational Options Pricing. *Bell Journal of Economic and Management Science*, 4:141–183, Spring 1973.

[10] C. Ng, D. C. Parkes, and M. Seltzer. Virtual Worlds: Fast and Strategyproof Auctions for Dynamic Resource Allocation. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 238–239, 2003. Short paper.

[11] N. Nisan. Bidding and Allocation in Combinatorial Auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, October 2000.

[12] N. Nisan and A. Ronen. Computationally Feasible VCG Mechanisms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, October 2000.

[13] D. C. Parkes, R. Cavallo, N. Elprin, A. Juda, S. Lahaie, B. Lubin, L. Michael, J. Shneidman, and H. Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, 2005.

[14] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *HotNets*, 2002.

[15] R. T. Rockafellar and S. Uryasev. Optimization of Conditional Value-at-Risk. *Journal of Risk*, pages 21–41, 2002.

[16] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-To-End Arguments in System Design. *TOCS*, 2(4):277–288, Nov. 1984.

[17] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[18] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation*, December 2002.

[19] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems. In *Proceedings of the 1st Workshop on Real, Large Distributed Systems*, 2004.