

IMPROVING BUSINESS OPPORTUNITIES OF FINANCIAL SERVICE PROVIDERS THROUGH SERVICE LEVEL AGREEMENTS

Henning Mersch and Philipp Wieder
Central Institute for Applied Mathematics
Research Centre Jülich,
52425 Jülich, Germany
{h.mersch|ph.wieder}@fz-juelich.de

Bastian Koller
Höchstleistungsrechenzentrum Stuttgart,
70550 Stuttgart, Germany
koller@hirs.de

Gerald Murphy
Belfast e-Science Centre (BESC), School of Computer Science,
The Queen's University of Belfast,
Belfast, BT7 1NN, United Kingdom
g.m.murphy@qub.ac.uk

Abstract The calculation of the Implied Volatility of stock options is a computationally expensive process which in general exceeds the resources available at a customer's site. Financial service providers therefore offer the required Implied Volatility services, adapting dynamically their own resource consumption to the customer's demands. The success of such a business model relies on carefully negotiated and observed Service Level Agreements between the different parties involved. The NextGRID project, driven by the adaption of several business scenarios to next generation Grid technologies, has designed and implemented an Implied Volatility framework which applies dynamic negotiation of Service Level Agreements to improve the existing solution. In this paper we describe the business scenario and the different core components which we integrated to realise the Implied Volatility framework.

Keywords: Implied Volatility, Negotiation Protocol, NextGRID, SLA, UDAP

1. Introduction

The development of next generation Grid concepts and methods and the application of business scenarios to evaluate those are two the major research and development areas of the NextGRID project [7]. One of the business scenarios is the calculation of Implied Volatility parameters for stock options (cf. Section 2). Since previous experiments revealed the potential business benefits of the application of NextGRID's Service Level Driven Dynamics concept [8] to this scenario, we integrated and extended NextGRID components (cf. Section 3) to realise the NextGRID Implied Volatility Framework described in Section 4. This framework shows some differences compared to other solutions, but offers convincing arguments for other projects to already make use of it (cf. Section 5). The development so far is promising and encourages future investment into maturing the existing solution (cf. Section 6).

2. Business Scenario

2.1 An Introduction to Implied Volatility

Within the stock market, stock and stock options can be purchased. Stock signifies an ownership position within a corporation. Options represent an option to buy (in the case of a call option) or sell (in the case of a put option) a set amount of stock from/to a third party at a set price (the strike price) in the future (the maturity date of the option). An option is purchased from the third party and if it is profitable on the maturity date (for example, the strike price of a call option is less than the current value of the stock, allowing the holder of the option to buy the stock more cheaply than would otherwise be possible) it will be exercised - otherwise it will be left to expire.

When the stock market is open, stocks and option prices are constantly being updated. Stock options are normally priced using the Black Scholes model. This equation contains a volatility parameter which can not be observed in practice. There is a one-to-one relationship between the theoretical price of a stock option and its volatility. Unfortunately there is no closed form solution for implying the volatility from the stock option price. If the volatility is known, trades can be executed to take advantage of volatility spikes. The Implied Volatility must be calculated using a numerical method; a Newton-Raphson iterative process is normally used, which is computationally expensive. The peak rate of the option market is 120,000 prices per second.

2.2 Current Implementation

The current implementation of this service relies on software being deployed on a local machine which is then hooked into a constant market feed. The limitations of this service are on the number of options which can be monitored

at any one time. As the number of options monitored increases the processing power required to calculate volatility increases exponentially, rapidly exceeding available resources.

2.3 NextGRID's Financial Service Scenario

In the NextGRID Implied Volatility scenario there are four actors: the *Financial Customer*, the *Financial Provider*, the *Compute Provider*, and the *Data Provider*.

Instead of the Financial Customers having the software running on a local machine or on one supplied by the Financial Provider they will have access to a Grid Portal where they can search for financial services (Implied Volatility being only one such service). They may also browse what Data Feeds are available to supply these services. When a Financial Customer selects a service they will be offered a choice of available SLA's with the corresponding Financial Provider and will then choose the most suitable. The completion of the SLA negotiation will stimulate the Financial Provider to discover from the Grid the necessary Compute and Data resources and set up the corresponding Service Level Agreements (SLAs) with Compute and Data providers (see Fig. 1 for the SLAs being negotiated between the different actors). Once all the agreements are in place the necessary software will be deployed onto the chosen compute resource, the chosen data resource will be connected and the subsequent output data stream will be supplied to the Financial Customer.



Figure 1. SLA relationships between parties

One essential customer requirement in the given scenario is high reliability, which implies the fast and accurate response of the system to failure. As an example, one could imagine the failure of the (or one) resource provided by the Compute Provider. In the current implementation this usually would mean either significant down time or the switch over to a shadow resource which has been mirroring the primary service. In this scenario the failure of the compute resource will be detected and the SLA between the Financial Provider and the Compute Provider will be breached. The Financial Provider will initially halt the supply of the data feed, discover a new suitable compute resource, agree a

new SLA with the new Compute Provider and deploy the software. The data feed may then be re-started pointing at the new compute resource which will then begin supplying the Financial Customer with the output data stream. From the point of view of the customers the SLA breach will only affect them in terms of a short delay while the Financial Provider switches Compute Providers.

The main driver for designing and implementing a framework (cf. Section 4) to fulfil the requirements of the scenario are the business benefits for the different actors: Financial Customers can now operate through a single access point (the Grid Portal) with a large range of services and smaller customers have access to a market not previously available. In case of the Financial Provider new revenue streams are created, specialisation in providing financial services not hardware or data expertise is now easier, and potential penalties due to breaching an SLA are reduced or prevented. And also the Compute and Data Providers can create new revenue streams because of the possibility to dynamically provide services to new Financial Providers.

3. The NextGRID Solution to dynamic SLAs

The two core components of NextGRID's Implied Volatility Framework are the *NextGRID SLA Framework* and the *Universal Dynamic Activity Package* (UDAP), which are both central entities of the NextGRID architecture [8]. The following two sections describe these components and their contributions to the architecture outlined in Section 4.

3.1 NextGRID SLA Framework

During the runtime of the NextGRID project a negotiation framework for Service Level Agreements has been developed [3]. The design of this framework is based on requirements from industrial applications which have been collected and analysed within the project. Based on this design, a proof-of-concept SLA Negotiation Framework implementation has been realised which is used to validate NextGRID's concepts and principles. This framework together with UDAP is the backbone of the Implied Volatility Framework.

Fig. 2 shows the design of the SLA Negotiation Framework. The two core components are Negotiators located at each of the business parties, the Financial Customer and the Financial Service Provider. These Negotiators are directly communicating and are responsible for Service Level Agreement negotiation in the NextGRID SLA Framework. NextGRID has chosen the so-called *Discrete-Offer-Protocol* for negotiation [8], which does not intend any refinements of negotiation parameters. In the beginning, the Customer Negotiator sends a request for an offer (a bid) to the Service Provider Negotiator. A bid has the same structure as an SLA template, but with empty information tags (service provider details, price, etc.). The Service Provider Negotiator has to check

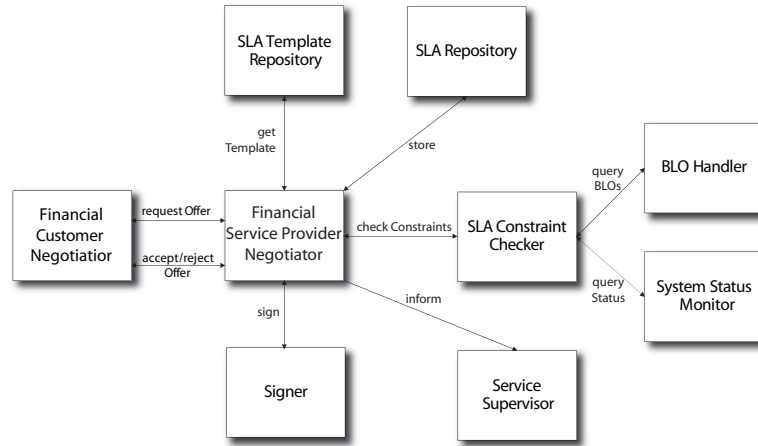


Figure 2. The NextGRID SLA Framework

whether or not it is able to provide the requested service. For that purpose it checks the SLA Template Repository (where Service Providers store their own service descriptions) for matching templates. If such a template exists, it has to be checked whether the current system status allows to offer this service to the Customer. The respective component – the SLA Constraint Checker – retrieves this information based on the selected service template from the System Status Monitor and from the BLO Handler. The BLO Handler is the component that has knowledge about the business preferences of the Service Provider (e.g. "prefer customer A to customer B", "maximise profit", "maximise resource usage", et cetera). Based on this knowledge, the SLA Constraint Checker validates the possible offer, and, if possible, advises the Service Provider Negotiator to fill the Customer's bid with the required information and send it back as an offer. Now the Customer has to decide whether to accept or reject an offer. In case of an acceptance, the Service Provider is informed and retrieves the SLA offer signed by the Customer. To become a valid SLA, the SLA offer has also to be signed by the Service Provider and a copy has to be sent to the Customer. Now each party has an SLA document signed by both parties. This SLA is stored in the SLA Repository component. After the negotiation process, the Service Supervisor component on the Service Provider's side is informed that a new SLA has been agreed upon and it starts to configure the system and the service(s) accordingly.

Please note that not all components of the framework have been implemented yet, since it was important to check the validity of the negotiation protocol first, a step that was possible with just a subset of the components.

3.2 UDAP

The Universal Dynamic Activity Package brings a new uniformity and coherency to managing activity information on a Grid. In the current state of Grid architectures, information about activities is fragmented and dispersed. Activity information, such as resource usage, security data, activity state, data requirements, et cetera, is currently captured using a variety of schemata and it is stored in different ways and by different logical components. This dispersion of activity information leads to management, security, and logistical overheads in discovering, accessing, and using that information. UDAP aims to bring all of the information fragments that are associated with an activity, regardless of the various schemata that are used to describe and capture these fragments, into one logical package.

The core of the UDAP model is the UDAP Document defined by the UDAP package, which is not discussed in detail here. The UDAP Manager is the entity that manages information in the UDAP Document. The UDAP Manager should have a standardised public interface that allows any Grid component to invoke its management functions for read, update and append operations of activity information. A UDAP Client is any Grid component that uses and/or produces activity information. UDAP Clients invoke the public management interface of the UDAP Manager for read, update and append operations of activity information. A UDAP Client can subscribe to the interface of a UDAP Manager, in order to receive notification of activity information based events. The subscription of a UDAP Client may be conditional, where the condition dictates the type of activity information based event that the client is interested in, e.g. "notify me if the state of the activity changes to running" or "notify me if the resource usage of the activity has exceeded the budget of the activity owner". An overview of the component's interactions is provided in 3.

3.2.1 UDAP applied to Implied Volatility. In this paper we will apply UDAP to the presented scenario for two purposes. On the one hand, Compute-, Data, and Financial Service Providers will be *discovered*. On the other hand, we will track a negotiated SLA to *evaluate* Providers (their QoS) as well as Customers. This means that UDAP is independent of the provider and the customer and lives in a separate third domain (see 3; different colours illustrate different domains). Nevertheless for other scenarios, the UDAP package could be used in other ways, which allows UDAP to participate either on the customer or provider side.

3.2.2 UDAP SLA Evaluation. If a UDAP user discovers a list of providers suitable for its purposes, it might be not enough to simply select the fastest or cheapest one. The UDAP user might want to order the providers additionally based on experience. Either its own experience or experience other users have

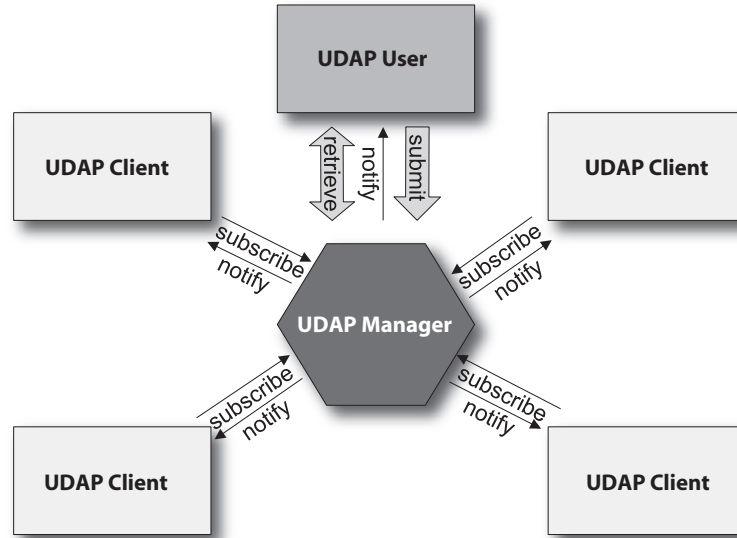


Figure 3. UDAP components and their interaction

done with. The same yields for the provider, which might want to make different offers (SLA Templates associated with the discovery result) depending on experience with this particular user. We call this the Quality-of-Service (QoS) of a provider respectively customer. For determining the QoS of a party, an evaluation system should be capable of tracking various SLAs, which belong to one party. This means this component could appraise a party depending on the history of its negotiated SLAs and associated QoS information.

Within the UDAP discovery step, it might be required to negotiate a SLA before consuming this service. UDAP could be used to track these negotiated SLAs for later evaluation. A SLA Activity Tracking instance could be instantiated by either the customer or the provider of the service via the UDAP Manager providing the negotiated SLA. Afterwards, the other party has to be informed about the returned EPR. Now both parties are in the position to append new information to this SLA Track called "Rankings". These items contain an overall integer representation of the satisfaction of the other party's QoS; additional statements of the other party like "violated an SLA" or "has not paid the bill" could be appended, too. The evaluation component could later on query UDAP to return all SLAs, where a particular party is involved. This brings the evaluator in the position to appraise a party by building the average over the provided satisfaction values. So, if a customer would like to order a list of providers, it could simply go to the evaluation component and retrieve the required information.

Some providers and customers would not like to publish their contracts anyway, because they are angry about the included information. This is fine for the presented evaluation system. If partners reject to support this system, no assumptions about their SLA will be tracked and the SLA would not be taken into account for evaluation.

4. Architecture

The NextGRID Implied Volatility Framework design aims to harness the Grid to allow fluctuating demand to be met by available resources on the Grid. In this section we describe and picture (cf. Fig. 4) the architecture of this framework.

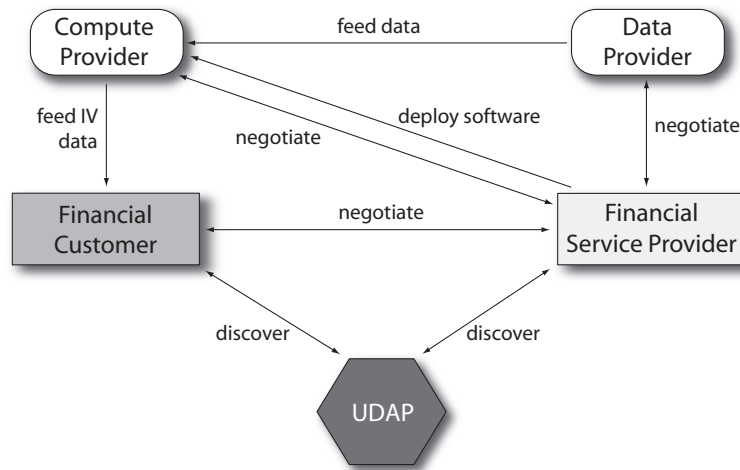


Figure 4. The Implied Volatility Framework

A Financial Customer is interested in retrieving a continuous data feed containing their interested information. To achieve this, it starts with discovering a service, providing this capability. This is done using the above described UDAP discovery facility. Afterwards, the customer retrieved a list of Financial service providers, which could be ranked by contacting the Evaluation system (not shown in the diagram). This uses prior agreed and tracked SLAs to rank the provided list. After choosing one Financial Service Provider, the the customer needs to establish a SLA with this provider by contacting its negotiator part. This will - if a SLA is established - check, if additional compute and/or data resources are required for fulfilling the requirements of the Financial Customer and will negotiate new Compute and Data Providers by discovering via UDAP and negotiate with the Negotiator Components of the providers. New compute resources need to have the software, which will be used afterwards, which

means the software must be deployed in a way that it could be used. Afterwards, the Data Provider could support the Compute Provider with required data for computing the Financial Customer required data feed.

4.1 Architectural Components

The **Financial Customer** is interested in an Implied Volatility data feed, which is provided by a Financial Service Provider. Therefore the customer first needs to discover a suitable Financial Service Provider using UDAP and then negotiate an SLA with the provider to agree upon the service level.

UDAP, as described in Section 3.2, provides two functions for our system: (i) discovery of different service types and (ii) the tracking of an SLA.

The **Financial Service Provider** requires compute and data resources to provide the Implied Volatility service (or other financial services). Both of them have to be discovered from UDAP and afterwards an SLA has to be negotiated with the Compute and the Data provider respectively.

A **Data Provider** provides a data feed which is the foundation for the Implied Volatility calculation.

A **Compute Provider** provides resources for computing the Implied Volatility data feed out of the data feed provided by a Data Provider.

4.2 Dynamic (Re-)Allocation

Normally the failure of a compute resource implies that the SLA between the Financial Service Provider and the Financial Customer is breached. In our architecture, the Financial Service Provider could - after detecting the failure - simply discover new compute resources with the help of UDAP, negotiate with the respective Compute Provider, and set up the machine by deploying the necessary software. Afterwards this compute resource replaces the old one, the Implied Volatility data feed is re-connected and the negotiated SLA with the Financial Customer has (ideally) not been breached.

5. Related Work

Apart from NextGRID a number of other projects is doing research in the area of Service Level Agreements in business-oriented Grid environments. AssessGrid, for example, introduces risk assessment and management to SLA negotiation [4], implementing, contrary to NextGRID, the WS-Agreement specification [1]. The same specification has been used by the Japanese Business Grid which provided solutions to share IT resources based on SLAs among distributed centres in an enterprise and trusted partners' data centres, thus making it possible for an Application Service Provider to dispatch a complex job through a single portal [5]. The reason for NextGRID not to use WS-Agreement, which combines a vocabulary for SLAs and a protocol to offer and negotiate

them, are the project's requirements regarding the SLA vocabulary. NextGRID supports the concept of self-similar SLAs [6], but WS-Agreement does not, although the protocols of both approaches are the same. The BREIN [2] project, on the other hand, develops solution based on the NextGRID SLA Framework and is therefore examined closer in the following section.

5.1 BREIN

BREIN [2] aims at realising flexible Virtual Organisation support to reduce the complexity of business-to-business collaborations. As already mentioned in Chapter 3.1, NextGRID concentrates on the conceptualisation of the framework, delivering reference implementations of selected parts of the framework. In the BREIN project, conceptual ideas of a set of projects (including NextGRID) have been taken up to develop an initial design of a prototype implementation of an "intelligent" negotiation framework.

In contrast to the NextGRID approach, BREIN wants to support a multiphase negotiation protocol. The customer sends an offer to the Service Provider Negotiator, which extracts the offer parameters and hands them over to an optimisation component. This optimisation component retrieves the business goals/business criteria of the Service Provider and the capability information for the resources as well as their availabilities. Based on this information, the optimisation component refines the parameters, hands them back to the Service Provider Negotiator, which then sends a counter-offer to the Service Customer. To automate these functionalities, BREIN will make usage of technologies and concepts from the Semantic Web and multi-agent area. At the point in time of writing this paper, the prototyping activity was not finished yet, which means that the presented approach is only a snapshot of the current status.

6. Status and Future Perspectives

In the course of this paper we described a business scenario, the provision of Implied Volatility services, which is used in the NextGRID project to drive the architectural development and to evaluate prototype developments. This scenario has been implemented integrating and enhancing NextGRID components, notably the NextGRID SLA Framework and UDAP (plus some auxiliary services that are of minor importance to this work). The resulting Implied Volatility Framework underpins the benefit of dynamic SLA negotiation to the actors involved in the business scenario. It is planned to demonstrate the function of the framework at different occasions. Moreover, the uptake of the concepts and prototype implementations by other projects like BREIN is ongoing work which will lead to more complete and advanced implementations of what has been presented here.

Acknowledgments

This work has been supported by the NextGRID project and has been partly funded by the European Commission's IST activity of the 6th Framework Programme under contract number 511563. This paper expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this paper.

References

- [1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. WS-Agreement - Web Services Agreement Specification. Technical report, Open Grid Forum, May 2007. Grid Forum Document GFD.107.
- [2] BREIN – Business objective driven reliable and intelligent grids for real business, 2007. Web site, 16 Jun 2007 <<http://www.eu-brein.com/>>.
- [3] P. Hasselmeyer, H. Mersch, H.-N. Quyen, L. Schubert, B. Koller, and Ph. Wieder. Implementing an SLA Negotiation Framework. In *Proc. of the eChallenges Conference (e-2007)*, The Hague, The Netherlands, October 24–26, 2007. To appear.
- [4] M. Hovestadt, O. Kao, and K. Voss. The First Step of Introducing Risk Management for Preprocessing SLA. In *Proc. of the IEEE International Conference on Services and Computing 2006 (SCC'06)*, pages 36–43. IEEE Computer Society, 2006.
- [5] H. Ludwig, T. Nakata, O. Wäldrich, Ph. Wieder, and W. Ziegler. Reliable Orchestration of Resources using WS-Agreement. In *Proc. of the 2006 International Conference on High Performance Computing and Communications (HPCC-06)*, volume 4208 of *LNCS*, pages 753–762. Springer, 2006.
- [6] P. Masche, P. Mckee, and B. Mitchell. The Increasing Role of Service Level Agreements in B2B Systems. In *2nd international conference on web information systems and technologies*, Setubal, Portugal, April 2006.
- [7] NextGRID – Architecture for Next Generation Grids, 2007. Web site, 16 Jun 2007 <<http://www.nextgrid.org/>>.
- [8] D. Snelling, A. Anjomshoa, F. Wray, A. Basermann, M. Fisher, M. Surridge, and Ph. Wieder. NextGRID Architectural Concepts. In *Proc. of the CoreGRID Symposium in conjunction with the Euro-Par 2007*, Rennes, France, 2007. To appear.