Typesafe

Bringing Reactive Applications to the Java Virtual Machine

Managing energy with a Reactive application

Ensuring that the energy grid functions correctly is a complex task requiring both consumers and producers to work in harmony to maintain required stability. Viridity Energy has built an enterprise software application that hides the complexity of the energy markets, links customer energy usage and market price data and provides clear, actionable information to its customers that can dramatically reduce their energy costs.

About Viridity Energy

Viridity Energy's innovative energy optimization solutions are helping energy consumers take control of their energy spend. Viridity Energy provides energy consumers with predictive optimization software that optimizes energy consumption assets with both energy market products and existing supply-side commitments. Viridity has a best-in-class ability to forecast market power prices and then monetize the available opportunities by factoring in and then leveraging all energy consumption assets. Ultimately, Viridity enables customers to effortlessly align their energy management with their business operations so as to maximize the available energy revenues and savings.

The Business Problem

One of the most significant challenges in the electric industry is the notion of keeping the electric grid stable. Supply and demand on the electricity grid must constantly remain in balance. On March 15, 2011, the Federal Energy Regulatory Commission (FERC) issued Order 745, which formally recognizes that consumers who reduce or manage their electricity prices in response to grid requirements actually help to achieve that balance just as much as generators that increase their production. As directed by that Order, electricity users who manage their electricity consumption in response to wholesale power prices are now to be paid at the same rate as electric generators. This Order represented a significant step forward in the evolution of electricity markets—and an opportunity to create new technology solutions that enables consumer participation in the electricity markets.

info@typesafe.com

Revision 1.0

As a result of FERC 745, Regional Transmission Operators (RTOs) across the country launched incentive plans to help stabilize the grid. Through multiple types of demand response programs, customers are actually paid to take certain actions so that they can minimize risk to the electrical grid and help keep it stabilized. The key behind an electrical grid is not to have spikes. It's very much like sound since there's a harmonic aspect to it. It needs to be kept stabilized, so it should not spike very high, or drop very low as doing so can actually cause physical damage to the components on the grid. Without this stability, the grid would fail to provide a stable, consistent supply of energy. So, to ensure dependable supply, there are different types of electric markets that are structured to achieve that balance.

In response to FERC 745, the RTOs expanded their energy markets – similar in concept to the Stock Market – where companies can bid. An accepted bid into the market will guarantee that certain generating assets owned by a company will be available for the grid to utilize, and the RTO will subsequently pay for that performance. With the post FERC 745 expansion, consumers can now bid in to the markets and reduce their electric load in response to grid needs. These same consumers then get compensated for doing so just like a generator. Essentially, large energy users with variable load can now trade "negawatts" (energy reductions) for payments in the wholesale Demand Response markets. There are three basic types of energy markets that create these opportunities:

- *Reliability Markets*, which compensate consumers for committing to avoid using a specified amount of electricity during times electric capacity may not meet regional reliability requirements;
- *Economic Markets*, which coordinate the commitment and dispatch of generation and demand resources and facilitate electric energy trading; and
- *Ancillary Markets*, which act as an insurance policy against the unforeseen loss of a major power plant or transmission line. In addition, they help balance the minute-to-minute flow of electricity.

It is important to note that different energy markets have programs that are either regulated or deregulated. Regulated programs will be primarily costs savings-oriented, while deregulated programs have demand response markets where the RTO will pay customers for managing their energy load in response to grid requirements. Consequently, some consumers are not eligible to participate in demand response markets as they are situated in regulated regions.

While new regulations brought a very rewarding opportunity to energy consumers; in practice, enabling a consumer to use electricity in a new way is a massively complex task.

The Solution

As the markets evolved, Viridity Energy realized that there was an opportunity to create a software solution to enable consumer participation in the smart grid world. By enabling two-way digital information exchange and creating a big data solution, Viridity would be able to collect massive amounts of data about how customers use electricity, and then distill that data into actionable information that enables those customers to actively participate in demand side energy management.

When Viridity was launched, they focused on optimization functionality, and built a software solution - VPower™-that analyzes the "digital footprint" of a company's energy assets. Through a proprietary optimization process, VPower™ determines the best way for a company to run their energy assets so that they can schedule them into the market at a time that does not compromise any business or operational constraints.

The way that Viridity does this is to analyze the two major aspects that affect electricity usage:

- Weather forecasts
- Energy Market price forecasts

Then, Viridity looks at the individual customer assets and the actual constraints associated with those assets and runs these variables through its optimization algorithms and modeling environment to produce a set of schedules that determines how best to operate those same assets. These assets are then bid into the market and generates a revenue stream not previously available. Through VPower™, customers have a platform to not only understand energy pricing in the power markets, but also have the ability to make informed decisions based on price and without compromising business constraints.

Implementing the Solution

Duncan DeVore joined Viridity as Chief Architect in March of 2010, and now is the VP of Engineering. He was tasked with designing the software that would implement Viridity's intended product offering. Duncan had experience with Scala from his prior position and, understood that it could be an excellent fit for this particular business problem. However he decided that Java coupled with the more traditional Spring MVC stack and Maven as a build system, would be a better way to meet the needs of the optimization centered platform.

After successfully launching VPower™ with Spring, Maven, and Hibernate, Viridity began actively working with clients in the marketplace. While the stack suited the needs of the initial intent, Duncan began to plan for the next stage in Viridity's commercialization—achieving an enterprise level SaaS platform with a cloud supportive architecture. In forecasting the future demand that would be managed by the platform, Duncan and his team began to examine solutions for the elasticity and resilience challenges that would eventually come into play. To rapidly scale the platform to meet future customer needs and after a year of evaluation and investigation into different stacks, it was apparent to the VPower™ technology team that the time had come to migrate to Scala. Scala is a general purpose programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It enables Java and other programmers to be more productive as it smoothly integrates features of object-oriented and functional languages.

Duncan began to slowly introduce Scala into the Viridity environment. - The initial effort focused on the optimization module as it was a highly concurrent [curve] set and by utilizing Scala and the Scala Actors library with the supervisor pattern (as this occurred prior to Akka becoming mainstream). The result was that the team was able to create a stable curve set that could handle optimization effortlessly.

However, not having a truly immutable domain model was an inhibitor to achieving scalability for distributed applications. This was an artifact of using the Hibernate model, and, consequently, not possible without a lot of re-architecting. Over a period of six months, Duncan formulated an architecture that he felt would address Viridity's business needs. The next step was validating this architecture with the vendors of the software products he intended to use in the new architecture.

The first vendor was MongoDB, and after spending time with one of their lead architects, Duncan felt confident in his choice of database. In particular, the challenges of big data are met by MongoDB. It provides a foundation for these systems - not only as a real-time operational data store but also in offline capacities. Additionally, MongoDB's native Aggregation Framework provides an extremely powerful tool set for predictive Data Analytics, which is fast becoming a central theme in the energy industry.

The next vendor was Typesafe, represented by Viktor Klang, former Team Lead for Typesafe's Akka project, and now Typesafe's Chief Architect. Akka is a runtime for building highly concurrent, distributed, and fault tolerant event-driven applications. After evaluating Typesafe's approach with his team, Duncan felt he would be in position to take VPower™ to the next level—to build a distributed reactive application backed by big data while reducing overall development costs. In making the decision to work with Typesafe, Duncan wanted to spend time with the lead for Typesafe, and walk through Viridity's approach. Viktor spent extensive time with Duncan, who walked him through Viridity's assumptions and approach. Viktor's impression, much like the MongoDB architect, was that Duncan and his team knew what they were doing, were taking the right approach and offered some suggestions on how Viridity could optimize things even more. It was during these sessions, that Duncan outlined his desire for a truly immutable domain model. Viktor advocated event sourcing, which aligned with Duncan's vision and led to a review of Greg Young approach - 'Command-Query Responsibility Segregation (CQRS). Due the strategic implications of the decisions being considered, Duncan and his team spent significant time understanding CQRS and event sourcing and how it could apply to Viridity's goals.

The team validated that this was the ideal solution, and started working on building an event-sourced framework. The difference between Greg Young's .NET approach, and the solution needed when building a system with an Actor-based model was immediately apparent.

As it turns out, Actors solve a lot of the problems that, in the .NET solution, you have to do manually. The whole process of collision is a big issue in event sourcing if you're not using an Actor-based model. My traditional understanding of event sourcing overlaid with the Actor model took a little while to figure out, but then I was introduced to the open source project Eventsourced. I looked through it and thought that this was both transformative for our architecture and strategically important for Viridity

Duncan DeVore
VP Software Engineering, Viridity Energy

A follow-on session with Viktor confirmed Duncan's evaluation and the team began building in earnest. One downside, that almost immediately surfaced was that Eventsourced did not support MongoDB, so Duncan, wearing his Chief Architect hat, took charge, joined the Eventsourced community, and built the asynchronous driver for ReactiveMongo himself - much to the delight of Martin Krasser, Eventsourced's founder.

An Elegant Architecture

Duncan's desire to build a modular architecture was realized through careful architectural choices. Sean Walsh, now Viridity Chief Architect, was tasked with resolving the asynchronous middleware layer utilizing Spray and Akka as well as security concerns and read side patterns, while Duncan focused on the backend Eventsourced components. Spray—another component of the Typesafe Reactive Platform is a toolkit for building REST/HTTP-based integration layers on top of Scala and Akka. Being asynchronous, actor-based, fast, lightweight and modular it's a great way to connect Scala applications to the outside world. Spray is a front-end to Akka, which now integrates Eventsourced natively (known as Akka Persistence). The application is completely optimized for writes, which are separate from the read side, where eventually

consistent data is stored in MongoDB, and then that data is projected to the front-end for consumption by users. The frontend is built with Angular.js, a common component in Responsive applications.

This model gives you Strong Consistency within the Aggregate Root (the event-sourced actor along with its entities) and Eventual Consistency in between. A design like this removes potential bottlenecks in the system by avoiding contention through automatic striping and is an enabler for linear—or close to linear—scalability.

Jonas Bonér,
CTO, Typesafe

Another beautiful aspect of this model is that no aggregation is required when retrieving data from the database; the data, in real-time, is continuously and appropriately formatted. Regardless of how complicated the application data being projected, the screen and response time is always measured in milliseconds. Complex real time graphs are generated constantly in the background, so there is no delay, when the screen loads.

This aspect is particularly important when it comes to big data management, a significant component of software that manages energy. This is an essential challenge that Duncan and his team anticipated as the business expanded and the SaaS approach was solidified.

The Reactive Manifesto

Jonas Bonér, Typesafe's CTO recently published "The Reactive Manifesto", a paper intended to define the blueprint for future enterprise applications. While Bonér wrote the first version, and leads the effort, The Reactive Manifesto is a collaboration of many thought leaders in the industry.

Duncan read the manifesto and was immediately captivated; the document defined all of the tenets that the Viridity application exhibited. Clearly, Duncan and his team are very proud of what they have been able to build with the Typesafe Reactive Platform, and are very excited that they have created a poster-child Reactive application.

- **Event Driven**: it's completely asynchronous since it leverages Akka for everything.
- **Scalable**: it's scalable thanks to Akka's configuration and ultimately to the Akka Cluster.
- **Resilient**: it leverages Akka's supervisor-based failure management to reliably handle failure.
- **Responsive**: the UI is incredibly responsive and interactive with the user because:
    - A professional design company designed the User Interface
    - It is built on top of Scala, Akka and Spray, which allows it to be responsive.

All of the aspects of a Reactive Application are there - they were there from the beginning - and it is a great testament to Duncan and his team for taking the time to build their application the correct way, leveraging the appropriate components from the Typesafe Reactive Platform to get the job done, so that they could concentrate more on building value into their innovative application.

When I read the Reactive Manifesto, it was like reading our technology roadmap, methodologies, and case for business expansion. Everything in this manifesto is consistent with our goal to create THE best-in-class total energy optimization platform.

Duncan DeVore
VP Software Engineering, Viridity Energy

*The **Typesafe Reactive Platform** is a modern software platform that makes it easy for developers to build scalable software applications. It combines **Play Framework**, **Akka** runtime, the **Scala** programming language, and robust developer tools in a simple package that integrates seamlessly with existing Java infrastructure. Commercial support and maintenance is available for the Typesafe Reactive Platform through the **Typesafe Subscription.***

info@typesafe.com

Revision 1.0