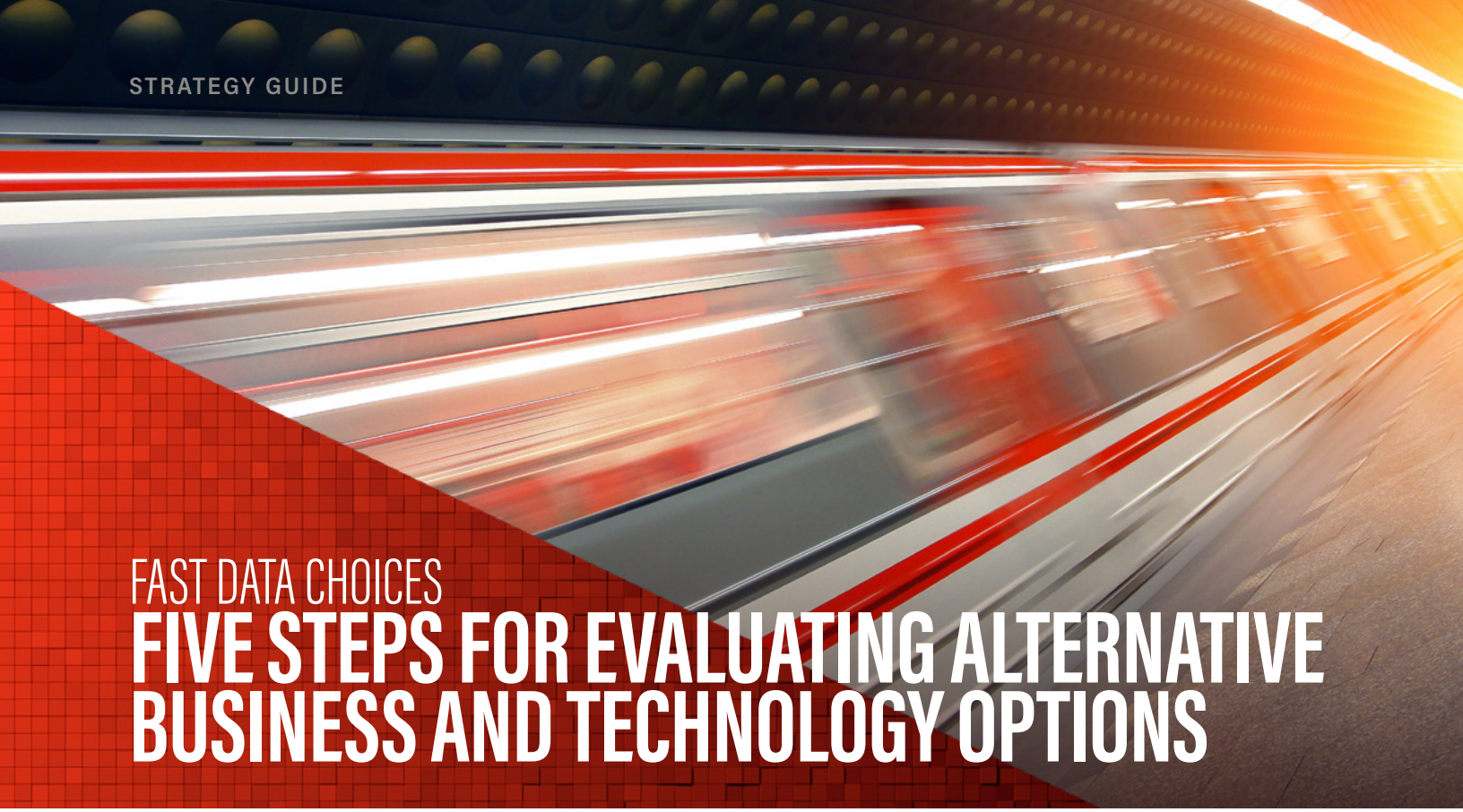

FAST DATA CHOICES

FIVE STEPS FOR EVALUATING ALTERNATIVE BUSINESS AND TECHNOLOGY OPTIONS

INTRODUCTION

Fast data is data in motion, streaming into systems from hundreds of thousands to millions of endpoints — mobile devices, sensor networks, financial transactions, stock tick feeds, logs, retail systems, telco call routing and authorization systems, and more.

Businesses are tapping Fast Data to be more relevant and valuable to their customers and using it to gain competitive advantage with more efficient and agile operations. Smarter customer engagement with real time, context-aware applications have direct impact on business results. According to Emagine International's CEO David Peters, "subscribers receiving tailored real-time offers bought 253% more services." Fast Data systems operationalize the learning and insights that companies derive from "big data".

Businesses are tapping Fast Data to be more relevant and valuable to their customers and using it to gain competitive advantage with more efficient and agile operations.

Telecommunications, mobile, advertising technology, financial services, and IoT sensor management are sectors benefitting from adoption of fast data strategies. But how can you move beyond strategies to results? Take a tip from management guru Peter Drucker, who said, "Strategy is a commodity, execution is an art."

This 5-step Guide to Fast Data Success will introduce you to fast data strategies at a practical level and provide you with a roadmap to rapid execution and implementation.

Here are the 5 steps we will walk through:

1. Identify your fast data opportunity
2. Assess and leverage your existing infrastructure
3. Understand the (business implications of) alternatives
4. Get agreement on success criteria for project
5. Prototype, pilot, refine

This can and likely will be an iterative process and the order of the steps is not absolute. While some of the steps have some dependencies on previous steps (you need to know what you will be attempting before being able to figure out how it will be judged to be successful), others can happen almost anywhere in the process. As you work through the steps, you might discover that there is not enough urgency or possible return on investment for the chosen opportunity. In that case, you go back to 1. and choose a new opportunity to pursue.

IMPORTANT NOTES

THIS IS ONLY A GUIDE

This is a strategy guide, not a complete and comprehensive step-by-step "how to" manual that will guarantee success. We have amassed a set of practical tips and we are sharing those tips with you here. This is intended as an aid for your own efforts rather than as a complete blueprint for a successful project. Should you need such a complete blueprint, please contact us and we will be happy to help you.

IT TAKES A TEAM

When we visit prospects and customers, we spend time with both the business side and the technical side. From the business people (Chief Marketing Officers, Product Managers, even Enterprise Architects), we get an understanding of the needs of

the business from a customer impact and financial perspective. They are the people driving to improve customer retention or to grow revenue or to expand into new markets. All of those business initiatives need technical implementations and support, and that comes from the technical people. To improve customer retention, you may need to reduce the latency in your customer interactions or to make additional data sources available in your reporting or predictive modeling to provide a more complete

(360°) view of your customer. To grow revenue, new products or offerings might be needed which involve increasing functionality or scope of applications. To expand into new markets, you might need to migrate to more efficient backend systems. The technical people are the ones who have to figure out how to do these things.

It needs to be a team effort. The business side needs to know what is practical and the technical side needs to know the business constraints for new projects. The new project won't be successful if business is trying to break the laws of physics (like wanting data communications at faster-than-light-speed rates). Likewise, it will fail if a newly designed system requires three times more resources, either equipment or staffing or both, than business budgeted for.

We suggest that business and technical teams participate together (and equally) in fast data projects.

1. Identify your Fast Data opportunity

Fast data challenges have common characteristics that make them easy to identify in the wild. These characteristics include the need for:

- ingesting unbounded streams of data
- analyzing the data stream in real-time
- taking some action on the live data stream
- exporting original data, as fast as it comes in, to a data lake or Big Data analytics engine. (This may also involve exporting enriched data as well.)

Fast data applications are categorized as such because responding in real-time to streams of data events it is critical to their usefulness and effectiveness. These are not simple look-up applications -fast data use cases need more than reporting or dashboarding; these applications need to 'act' by making decisions on live data feeds. Approve this credit transaction or flag it as potentially fraudulent; let this mobile call go through or alert the caller that they are out of minutes; serve a particular ad to a particular consumer visiting a particular web page; alert an investment broker of a critical risk exposure for the high net worth individual they are speaking with: all of these are real-world fast data applications we have helped our customers address.

Let's take a look at some common use cases for VoltDB — it's a sampling of ideas for how you can put fast data to work in your enterprise. They illustrate the applications of fast data at some of our customers and list some of the potential business benefits.

- **Personalization 2.0 (Hyper-personalization)**
  Know more about every customer (or potential customer) and fine tune each interaction to each individual based on their profile, their location, their actions, etc. Ad-tech companies, online gaming app developers, mobile service providers, and financial services firms use VoltDB to deliver more engaging customer interactions with richer and more relevant customized content and offers, which drives higher satisfaction and revenue within every interaction, all at enterprise scale.
- **Resource management and real-time policy enforcement (including fraud detection)**
  Telcos use VoltDB to manage the deluge of data from mobile, connected devices in real-time, enabling mobile service providers to act instantly to provide real-time call authorization; create new services and applications; improve quality of service; meet SLAs; detect and prevent fraud; and create better customer engagement for greater subscriber retention and ARPU — with millisecond response times.

- **Internet of Things (IoT) sensor management and data processing**
  Sensors for 'smart grid' electricity management, factory automation, and smart cities and smart homes produce high-velocity data that needs to be ingested, analyzed and acted upon in real time to produce business value. VoltDB processes fast data from more than 60 million sensors fielded by leading energy providers, and automates temperature and pressure control of gases used in semiconductor fabrication. Enabling these and other applications that deliver on the promise of the IoT in complex environments requires a simple, unified real-time database.
- **Real-time billing and bid management**
  Financial services organizations use VoltDB to ensure compliance with regulations governing best bid and offer pricing, also delivering real-time automated stock and securities recommendations to broker-dealers. AdTech companies take advantage of VoltDB's transactionality to solve the "last dollar problem" of accurately spending against their clients' full ad budgets. Telcos also use VoltDB to manage real-time call routing and billing authorization applications. Customers across industries use VoltDB to accurately and precisely manage budgets and inventory in real-time.

Get more detail about VoltDB Case Studies.

2. Assess and leverage your existing infrastructure

You've invested considerable time and money into your IT environment over the years and you have built systems that work well, quickly and reliably, for the jobs for which they were designed -it would be wise to try to continue to use those systems where possible. We see many customers who have extremely functional, reliable and efficient data warehouses, for example. These data warehouses are responsible for supporting important business intelligence efforts like reports and executive dashboards, tools that are core to the functioning of your business so it could be unnecessarily risky to consider replacing those systems. They can also provide the ability to perform deep or predictive analytics, often a complementary function to fast data applications.

Start by looking at your existing infrastructure and evaluating what can be adapted in your quest to create business advantage with fast data.

Everything should be made as simple as possible, but not simpler

Complexity is a drain, on finances as well as on time and resources. A simpler solution is easier to implement, easier to maintain, easier to live with. But make sure you don't try to simplify too much -you can lose performance or functionality if you do. Swiss Army knives do many things but often not as well as single, standalone tools. They represent a compromise -you give up the full functionality of each of the single tools for the convenience of being able to carry them all around in one compact tool.

We see the same thing happening in IT. There are vendors that say their product is a single solution for two different problems. Hybrid Transactional and Analytical Processing (HTAP) (from Gartner) and Translytics (from Forrester) are great concepts and wonderful goals, but current offerings are proving to be something less awesome for both analytics and transactions -they tend to be good or even great at one and marginal on the other. Sometimes it is better to use two best-of-breed products that have tight integration between them rather than settle on a single product, especially if you already have one of the best-of-breed products and it is working well for you.

VoltDB has fast ingest and export capabilities, allowing it to integrate seamlessly, with minimal latency, to existing data warehouse and data lake platforms like Teradata, HPE Vertica, Cloudera, Hortonworks, MapR, and IBM Netezza, among others, directly or through queueing systems such as Kafka, RabbitMQ, and Amazon Kinesis.

3. Get agreement on success criteria for the project

What does fast data success look like for the defined project?

Every successful development project begins with a success profile built on user requirements, project scope, and measureable success metrics.

Chances are good, especially if you're in a data-driven business, that your developers and architects are already thinking about — or implementing — a fast data approach. But do they know the scope of the problems the enterprise is trying to solve? Have they gathered user requirements and business constraints from all stakeholders? And how will you and they, together as a team (remember?) know when success is achieved?

Continue your journey to fast data success by walking through these steps:

- Describe the project's primary purpose and characteristics (see section 4 for more details on these):
  - Is the system or application analytic or transactional?
  - What are the requirements for accuracy and consistency of data?
  - What is the uptime/availability profile?
  - Is the app or system customer facing or in the revenue path?
  - How quickly must the system or app be implemented?
  - What languages, standards and design approaches are a good fit for your needs?
- Scope the project. We don't intend for this to be a complete prescriptive guide for project development and execution, but here are some helpful points that we've learned through working with our customers on fast data projects:
  - Narrow your candidate list down to perhaps three possible use cases and have your team agree on which is the most pressing -which will provide the most "bang for the buck", not only for money and time invested but also the lost opportunity cost in not pursuing the other use cases. That will tell you what your first fast data project should be. Break the project into sub-projects and assemble teams with domain experience in each. If you pursue agile development (as most fast data project teams do), these will be your scrum teams.
  - Build a timeline (a suggestion: read Elihu Goldratt's Critical Chain or Fred Brooks' The Mythical Man-month first)
  - If the project is for changing a current customer-facing system or app, you'll need to make sure those systems/apps keep working so your pilot cannot impact existing systems.
  - Make sure your development includes development of test harnesses and that you include a QA team in the planning and execution.
  - Choose technology components that are true solutions: If performance is a bottleneck, set throughput and transaction per second goals. If scalability is an issue, determine if you need

scale-up or scale out and choose software that supports that decision. If data accuracy and consistency are critical, look for an operational system; pure analytics systems are good for insight but can't deliver accuracy or consistency.

- Define success:

  Using the pain points that drove your choice of this project, build a success profile that shows how you can make progress resolving persistent issues, including

  - » poor performance
  - » lack of scalability
  - » importance of data consistency
  - » need for real-time response and low latency
  - » how to simplify development of apps that are easier to test and maintain
  - » requirements for ecosystem integration
  - » readiness for cloud deployment
  - » Fully document the agreed upon success metrics so the results of the testing can be accurately assessed.

- If at all possible, get executive approval (and sponsorship) for the chosen use case. Having an executive with "skin in the game" will help you push through when issues arise, particularly when budget is involved. This is usually easy to do for fast data applications as they are often operational in nature and directly impact key financial metrics — executives are eager to support projects that will positively impact their bottom lines.

4. Understand the business and technical implications/impact of your solutions alternatives

You've identified a fast data opportunity and scoped out the success metrics. Now it's time to choose technologies and consider the implications of the alternatives.

In our experience working with hundreds of developers on many different fast data use cases, we've learned there are some key questions and considerations that help you identify fast data project(s) that will be successful:

- **Are you addressing an analytics problem or transactional problem, or a combination of both?** Is this an operational problem? The definition of operational used to be clear -transactional systems were operational and analytical systems were not. Recently, though, the line has been blurring and some companies have moved their analytics into the operational realm. Analytics applications and systems are often termed online analytical processing (OLAP), geared to insight and exploration. If you want to know about things after they've happened or dig deep into who/what/when, an analytics solution may work well for you. But if you're dealing with applications that sit between your business and its customers, you are solving an operational problem. There are many alternatives for analytics, but few for real-time analytics, and only one system for real-time analytics that support operational workloads via transactions on live data, also known as online transaction processing (OLTP).

- **Is it real-time or batch?** Most fast data problems are operational but this is a good gut-check question to make sure: Would batch-oriented processing provide results that were "good enough"? It also helps to define real-time — different industries and use cases have different interpretations. We tend to gauge real-time in terms of microand milli-seconds latencies and response times.
- **Is data integrity important?** Other ways to ask this: Is your data important? Is it ok to lose some data? In distributed systems, consistency is an important consideration.
- **Some data is less valuable than other data;** if slightly different values are returned to different clients, or even if we lose some of it, there is no real harm. For that kind of data, the eventual consistency offered by most NoSQL solutions might be fine. But for financial data, any applications that do billing or accounting of any kind, exact, consistent and complete data is critical and fully ACID-compliant transactional systems should be on the top of your list. Business people often don't realize the tradeoffs on data consistency that their dev teams make in creating fast applications. Be sure to have the open conversation with them -we have heard of some serious financial (and security) lapses in applications when using an eventually consistent system where full consistency was really needed.
- **Batch vs. real-time (now or later):** Can you afford to wait an hour, six hours, a day for answers from your system? Do you need to take action in realtime, for example to authorize a mobile call; serve an ad; personalize an offer; or evaluate a card swipe for fraud? Are you building a customer-facing application which needs millisecond response times in order to keep the customer engaged? If so a fast operational DB may be the simplest, easiest approach. (Check out the chart Mike Stonebraker used to describe this here.)
- **Purpose-built, best-of-breed vs. big Apache stack:** If you're solving an analytics problem, stitching a solution together from Apache projects like Storm and Spark may be sufficient. If you're working with operational data, you want a tested, purpose-built enterprise-class solution. Avoid the urge to write the glue code necessary to get what you need from the Apache stack: it's harder to write, more error-prone, and frequently untested. It introduces lots of risk. Maybe a message will be delivered, or maybe it won't. Maybe a processed value will be correct; maybe it won't. Can you afford the risk and uncertainty?
- **Conduct a reference check among your peer group and principal analysts.** The high-tech world is in constant flux and growth.
- **Evaluate existing skill sets:** Do you practice Agile, waterfall or a hybrid? What strengths do your developers have? Are you a Linux or Windows shop? Are you operating a virtualized environment, perhaps in the cloud? Do you value the expressive programming model of SQL, the ubiquity of Java? Play to your strengths.

5. Prototype, pilot, refine

It's finally time to work the solution. Many companies use the term "proof of concept" (POC) or even "proof of technology" (POT) to describe their trial process. We recommend that you think about it as project prototype or pilot. During this step, you should attempt to build a minimally viable product (MVP) version of your project provide strong evidence that the final project will be successful. POCs/POTs for arbitrary use cases using arbitrary data and data sources will not prove anything other than the product you are testing does something.

Working with the product in a real scenario like a prototype will also expose the "warts" that are often only discovered after living with the product for a while.

Be realistic about timeframe and resources. Many small companies have few resources to dedicate to prototyping, especially prototyping with several alternative products. Unless you have multiple teams to put on this, it may be completely impractical to play with more than one candidate product at a time.

Download software and test against your use case.

Many fast data solutions companies make it easy to try their products. Typically it is as easy as visiting their website. If they offer both commercial and open source versions, you will need to decide which to try. Companies usually ask for contact information before they allow you to download their software or to receive a license key. That registration information can and will be used against you sometime in the future -which means that you will get calls or emails from their inside sales team. Sometimes that is a good thing so if you are seriously considering a certain product, don't fret about giving out your digits.

Open Source version? Paid commercial version?

The decision between open source and commercial software is not as straightforward as you might first think. Your first thought might be that you like the "free" aspect of open source so that is what you want to try first. Sometimes you can even download open source code without any kind of registration. But some vendors have more functions/features in their commercial versions. Perhaps more important, they offer support with their commercial versions. VoltDB offers an open source version (available on GitHub) as well as an Enterprise licensed (commercial) version that includes support, so you get a choice. While you may think you don't need support at this stage, for an important project with a tight deadline, support might mean the difference between making and missing a critical deadline. Make sure you know what is and is not in the open source products and fully understand the implications of not having paid support on the product before you decide. Of course if you can change your mind later on, you can pay to get the full commercial version and support -vendors are usually very happy to convert you to a paying customer.

You already have a set of success criteria for your project (defined and agreed to in the previous step). Success of the project will be judged by this set of criteria so it is important that the benchmark fully and adequately tests that set of criteria to the required level. Critical factors like scalability can't be assumed or simply projected through calculation.

If the project was successful, the next natural step would be to put it into production. If it wasn't successful, you have some think- ing to do. Go back to your team and do a post-mortem. Some common problem areas you can explore: Was it just the wrong use case? Were success criteria unrealistic? Did you do an adequate assessment of possible candidate products? You may just need to refine your original project slightly or you may need to start over. Either way, you likely learned some valuable lessons and gained valuable insights from the experience.

Fast Data success — a review

Big Data, Cloud, streaming — these technology paradigms have driven massive investment, although quantitative benefits have been hard to document. Now that you know the problem you're trying to solve, you'll be positioned to choose the best fast data solution for your use case.

We've seen that many options exist, some skewed towards analytics and reporting; others operational in nature. You know if your challenge or use case is in your organization's revenue path, it's an operational use case — one in which hard-dollar benefits are simpler to extract — but there are other operational applications that don't directly touch revenue and so might be more difficult to pick out. Here's a quick refresher of the differences between analytics (OLAP) and operational/transactional (OLTP) use cases:

- **Reporting and analytics**
  Do you want to enable fast queries against raw (structured) data? Is organizing data at query time and performing queries across multiple dimensions of terabytes to petabytes of stored data the most important part of your application? If you answered yes to the above, you're solving an OLAP-style problem. But if you also need multi-factor decisions, choose a fast data solution that delivers both analytics and transactions. And remember what Turing laureate Mike Stonebraker says: Even if you don't need transactions now, you might want them later.
- **Capturing and performing computations on data streams**
  Some architectures are designed to capture fast streams of data. They aren't optimized to store data or do fast record lookup. Systems built to support these streaming applications often are good at scaling predefined queries against fast streams, optimizing scalable message processing and coordination between systems. Although they're good data pipelines, most achieve speed by sacrificing data consistency, correctness and ACID transactionality.
- **Per-event decisions**
  Architectures and applications that sit in the revenue stream of an organization are operational. These designs focus on enabling real-time analytics and per-event decisions on live streams of fast data while maintaining ACID semantics, data consistency and accurate answers.

Download "The Developer Guide to Streaming Data Applications" for more details.