Thumbtack Cloud-Ready Architecture

ABOUT THE THUMBTACK CLOUD-READY ARCHITECTURE

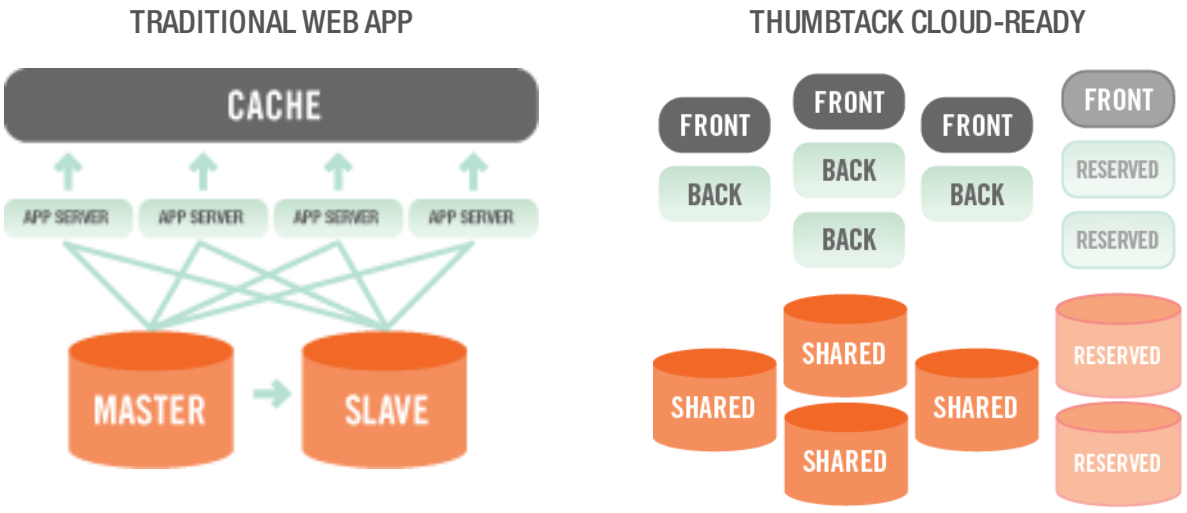
Thumbtack ensures that all software we deliver follows a rigorous set of architectural design guidelines. These guidelines ensure that applications are delivered in a "cloud-ready" state, whether or not they are intended for deployment to a public or private cloud in the short term. The architecture provides immediate benefits to all applications, regardless of deployment scenario, including:

- Development flexibility
- Lower cost to maintain and improve
- Simplified transition to internal development teams
- Fault tolerance
- Horizontal scalability
- Embedded testing

These are achieved by designing a system as discrete functional units, each of which has its own performance and quality metrics. These are built and monitored individually, as well as tested functionally as part of the whole.

ARCHITECTURAL OVERVIEW

The primary tenet of the Thumbtack Cloud-Ready Architecture is to divide the application into discrete independent services with no single points of failure. This is done throughout the application stack, and often involves steps such as pulling storage into distributed key-value stores (NoSQL databases) and providing direct service level guarantees instead of relying on complicated caching and recovery processes.



The cloud-ready example above provides several key advantages over the traditional web architecture. It breaks apart the app server layer into discrete components that can be independently provisioned and measured, reducing deploymoent costs and increasing redundancy. It also reduces the need to maintain everything in a monolithic cache, which can dramatically decrease startup time and improve application performance while the cache warms. By replacing the relational database infrastructure, the operational costs of maintaining the system are dramatically reduced, while the system on the whole becomes more scalable and a single point of failure is eliminated.

155 Water Street, Suite 512 Brooklyn, NY 11201
(718) 514-2063 sales@thumbtack.net thumbtack.net

FINE-GRAINED SERVICES

We design our systems as interconnected functional components, each providing a well-defined service or set of services. We view "services" as describing the design and interaction of the various components, not necessarily the protocol used to tie them together. The actual protocols used vary based on the specific requirements. For example, we often use REST-based APIs to provide a simple loose coupling between components, but in cases where low latency is paramount, services can be wired together in process or invoked asynchronously using a high-performance message queue such as ZeroMQ.

As with any architectural decision, there are positives and negatives associated with each design choice. For example, it often makes sense in start-ups to build systems as a monolithic bundle that provides all functionality to all developers. As companies grow, however, a number of benefits become immediately available as services are introduced:

- **Support for Independent Development Teams.** By defining hard and fast boundary points, the interactions between teams can be more easily made explicit. This reduces management overhead and helps scale development teams across geographic boundaries.

- **Quality of Service.** Each service has its own instrumentation, allowing quality of service to be monitored explicitly. Latency and throughput guarantees can be adjusted to each specific business process, and resources can be allocated more efficiently. Lower level integration tests become easier to produce.

- **Pluggable Implementations and Business Discovery.** By defining a business function purely in terms of its services, it becomes much easier to swap in alternatives and compare results. This not only enables a smoother transition during upgrades, it enables the business to discover "hidden" undocumented features that have emerged over time. For example, before switching tax calculation engines, we might run both systems in parallel for several months. If the two systems ever disagree, we've identified an undocumented use case we need to account for. (Or an unidentified bug we inadvertently fixed!)

More generally, by defining each technical process as an independent service, they become much cheaper to maintain and extend. Capacity planning is more transparent, and cost metrics much simpler to compute. The fact that everything is exposed in a similar manner minimizes infrastructure planning and operational overhead.

DESIGNING REPEATABLE PROCESSES

By "Repeatable Processes" we mean that each system is designed with the expectation that something that can go wrong at any time. In an era of increased virtualization and cloud computing, this requirement becomes more and more critical as failures become more common and less easy to identify. Traditionally robust systems such as disk access become less predictable in virtualized environments, and often fail in bizarre or unexpected ways. More ominously, they often fail in ways that are not immediately identifiable as failures, such as dramatically increased latency or partial reads. These are not new problems, but have become more central to application concerns as companies move increasingly to virtualized environments

The central way in which Thumbtack addresses these concerns is through what we call *Repeatable Business Flows*. This means in a nutshell that every system can be invoked in isolation at any point in time, and should either succeed or fail in a predictable manner. In the event of failures, data should not be left in an ill-defined state, and remediation steps should be clearly defined and automated. We achieve this through:

- **Idempotent Guarantees.** Wherever possible, we build services as idempotent. This means they can be safely invoked repeatedly if necessary without changing downstream behavior. For example, if we experience an internal failure after potentially billing a customer, we will ensure that the process can be retried without risk of double-billing.

- **Time Independence.** Scheduled processes are designed to function independently of the schedule they actually run. This allows for transparent recovery in the event of failures, typically without any manual intervention. For example, if a process runs nightly, but fails for some unidentified reason, we ensure that the process can be restarted as often as necessary at any time of time of day and produce the same results as it would at night.

- **Run-Time Guarantees.** Each system has its own monitoring and instrumentation, and makes performance guarantees to its clients. If a system does not provide a fully-formed response within its guaranteed timeframe, it is treated as a hard failure and processing continues. This allows fail-fast behavior in environments that might not provide that capability natively.Combining the above allows systems that not only recovery gracefully from external failures, it also allows us to build systems with very tightly-controlled availability guarantees. For example, we can assemble the components in such a way where multiple copies of a service are invoked in parallel, using only the copy that responds the fastest. This enables us to make trade-offs such as increasing the number of cloud nodes in exchange for more reliable timing.In general, the above principles can be combined in different ways to customize the predictability and fault-tolerance of any system we develop.Cloud Challenges

155 Water Street, Suite 512 Brooklyn, NY 11201
(718) 514-2063 sales@thumbtack.net thumbtack.net

Combining the above allows systems that not only recovery gracefully from external failures, it also allows us to build systems with very tightly-controlled availability guarantees. For example, we can assemble the components in such a way where multiple copies of a service are invoked in parallel, using only the copy that responds the fastest. This enables us to make trade-offs such as increasing the number of cloud nodes in exchange for more reliable timing.

In general, the above principles can be combined in different ways to customize the predictability and fault-tolerance of any system we develop.

CLOUD CHALLENGES

The architecture described above is appropriate for all systems that require a high amount of uptime with rigorous controls for speed and responsiveness. But it becomes absolutely critical for managing any application in the cloud.

The most difficult part of migrating any application to the cloud is to ensure the application is built to leverage the flexibility the cloud provides. In order to leverage the built-in scaling of providers like Amazon or Rackspace, applications need to be able to transparently distribute themselves among new nodes as they come up, which means the applications need to be built with the assumption that resources can come or go at any time.

Moreover, virtualized cloud environments expose applications to new types of environmental failures compared to traditional hosting. Services such as Amazon's Elastic Block Storage offer powerful tools to ensure highly redundant data, but also cause unexpected outages that are difficult to diagnose or even detect. In such circumstances, it is more critical than ever to have no single points of failure.

The Thumbtack Cloud-Ready Architecture is designed explicitly to address these two challenges of horizontal scalability and fault-tolerance, whether within or outside the cloud, on the assumption that eventually all applications will benefit from cloud-like services. Of course, no single paradigm is 100% appropriate for all scenarios, but we feel that the majority of public-facing applications can benefit from this approach.

CONCLUSION

Thumbtack applications have proven themselves again and again under high-stress environments. We achieve this success through a rigorous development and code review process, and also by internalizing development guidelines throughout our company. The first step of any engagement is to identify potential sources of trouble in the overall environment, and the last step is to perform rigorous stress testing and performance analysis to ensure the application works under any possible scenario.

Whether the application is intended for the public cloud, hosted internally, or in a hybrid combination, Thumbtack's ability to achieve the core goals of fault-tolerance and horizontal scalability allows our customers to maintain it at minimal cost and expand it in predictable and measurable ways.

155 Water Street, Suite 512 Brooklyn, NY 11201
(718) 514-2063 sales@thumbtack.net thumbtack.net