

Top 10 Uses of IronWorker

Background Processing, Event Handling, Scalable Distributed Systems, and More

Using Workers Will Change the Way You Think – And Program

First generation applications are often developed as simple two-tier systems. There is an application tier, which runs the application's software, and then a database tier which stores the application data. More serious applications, however, need to scale up from this type of architecture into something that's more capable and expandable.

This need is especially true for production-scale cloud applications where the load factor can be much greater and the pressure to meet customer needs more greatly accelerated than traditional applications. Modern app development almost dictates building a multi-tier architecture right from the start.

Multi-Tier Architectures and the Role of a Worker System

In a multi-tier architecture, the additional tiers most often include message queues, worker systems, and key/value datastores, to name a few. These components help applications improve app responsiveness and address scale by pre-calculating user data, pushing non-essential processing to the background, and providing asynchronous processing or event handling.

Worker systems provide the muscle in these types of distributed systems. They are the processing power that actually gets things done. Workers generally provide for custom-built environments that only survive for the duration of the task – whatever work needs to be done, like resizing a single picture. Then the environment is destroyed, and rebuilt for the next task. The benefit of workers are their flexibility. Because they're highly reproducible and self-contained, workers can be scaled effortlessly across VMs to meet demand.

The benefits include:

- Increased app responsiveness - processing can be scaled and prioritized
- Increased scalability - ability to seamlessly scale throughput and functionality
- Reduced complexity - reduced risk/overhead in critical but non-strategic areas

A worker system is a key part of any scalable architecture. It allows processing to be moved to the background, supports asynchronous event handling, and offers highly concurrent/ scale-out processing options.

The ability to push task-specific code in almost any language to the cloud and then run it reliably at scale opens up a new world of options.

We guarantee, once you realize the power of a scalable worker system, it'll be hard to go back to the old way of doing things.

Try IronWorker

Iron.io provides a multi language platform for running tasks at scale. Try it for free today:

www.iron.io

For additional information, contact sales at:

1-888-939-4623

Or by email at:
sales@iron.io

Top 10 Uses of IronWorker

1. Image Processing



Photos, ads, diagrams, and other visual information are critical pieces in consumer and business applications. They convey critical information and increase engagement. Nearly every use of photos requires some element of image processing whether it's resizing, rotating, sharpening, adding watermarks, or creating thumbnails. Image processing can be compute-heavy, asynchronous in nature, and linearly scaling (more users mean more processing). These aspects all make it a great fit for the elastic nature of IronWorker.

2. Document Processing



Document files are at the core of content sharing. They can come in many formats – .txt, .pdf, .doc, .xls, .ppt, and .html, to name a few. The type of operations that can be performed are dependent on the file type and the code libraries or API services available but if there's a way to process them, IronWorker can provide the workload engine. For example, converting a Word document to text or an Excel file to csv might use packages such as *catdoc* and *xls2csv*. Converting an HTML document to PDF might make use of something like *wkhtmltopdf*. Workers can be created to use these code libraries to run on IronWorker.

3. Web Crawling / Data Extraction



The web is full of data — from social, to weather, to real estate, to financial information, data is available to access, extract, share, create derivatives, and transform in any number of ways. But crawling and extracting data from the web requires lots of concurrent processes that run on a continual or frequent basis. This makes it another strong fit for background processing and IronWorker.

4. Push Notifications



Push notifications tend to go out in large sets. For example, a news alert might be sent to millions of subscribers. Sending these notifications out in serial batches can take too long. A better architecture is to use IronWorker to deliver these push notifications through APNS and GCM in parallel. This approach also lends itself to processing the lists on the fly to either dedup lists or offer customized messages.

5. "Any" Data Processing



In the end, a large amount of "big data" use cases essentially boil down to large scale "data processing" and IronWorker is made for this. Let's say you have a big list of zip codes and need to pull weather data from a weather API as well as population data from a different API which times out after 10 concurrent connections. Traditional "big data" solutions are simply too complex to manage these situations. IronWorker provides a flexible but still massively parallel way to accomplish this.

Top 10 Uses of IronWorker

6. Sending & Receiving SMS



It's a snap for developers to use SMS services to build these capabilities into the app but the process to check and send this price change alert is inherently something that runs in the background – which means another use case almost specifically designed IronWorker.

7. Sending Emails



A common pattern, for example, is to send a nightly report to customers. Say you have 1M customers. Generating this report might access your database, make use of 3-4 different APIs, and then finish by sending an email using SendGrid, Mailgun, or Mandrill. If each customer took 10 seconds to process, serial execution would take 115 days to send this report. Obviously, not something that can be done without some element of scaling. Enter IronWorker.

8. Replacing CRON



The scheduling capability within IronWorker is simple to use and offers greater reliability than CRON jobs. Developers can incorporate all of the processing within the scheduled task or they can have this task kick off other workers to divide and conquer larger workloads (a common pattern). All of the processing in this article can be done as a result of a direct user events or can be triggered on a particular schedule. Using IronWorker, you can eliminate the need to monitor or manage scheduling infrastructure.

9. Processing Webhook Events



IronMQ and IronWorker have native support for webhooks. IronMQ can accept webhooks as well as post to them. IronWorker also lets developers create their own webhook responses. By this we mean developers can upload a worker and then have it serve as an endpoint for a webhook. A task will run when the webhook is posted to. The beauty here is that there's no need to have an underlying application – just the code to process the event uploaded in IronWorker. Developers provide the event logic and IronWorker provides virtually unlimited processing capabilities.

10. Adopting a Service Oriented Architecture



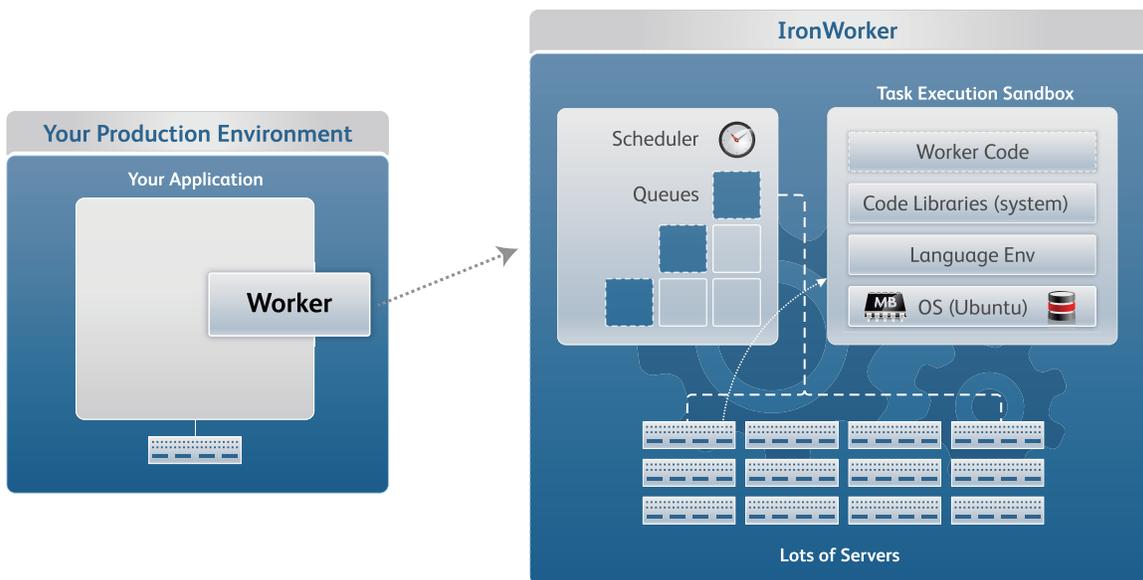
There is growing trend to shift from monolithic app structures and tightly bound legacy systems to more scalable, evented models. One where message queues are used to orchestrate asynchronous processing and where much of the processing is done by worker systems in the background independent of mainline processing.

IronWorker

Production-Grade Worker Platform

Iron.io provides a production-grade worker platform called IronWorker. It is a task processing and scheduling service that provides a powerful way to make applications more responsive, increase development speed, and gain instant scalability. IronWorker supports all common languages including binary packages and offers task retries, hooks for real-time logging, sophisticated task monitoring, and more.

It is a scalable cloud-based service that operates in multiple zones for reliability and high availability. The use of cloud services creates efficiencies and agility because it eliminates having to stand up and maintain infrastructure components, not to mention making them redundant and failsafe. (Iron.io's offerings can also be installed in private clouds and even on premise, providing enterprise-grade and even carrier-grade capabilities. For more information on these options, contact our sales and support group.)



Visit www.iron.io today to start distributing instantly