EP59: 90% Cost Slash: From Serverless to Monolith

ALEX XU
MAY 13, 2023

109 10 3 Share

This week's system design refresher:

- Top 7 Most-Used Distributed System Patterns (Youtube video)
- Amazon Prime Video Monitoring Service
- RPC vs. RESTful
- The cost of storage systems

Meet New Relic Grok, the World's First Generative AI Observability Assistant (Sponsored)

New Relic Grok makes it easy for you to get the insights you need without having to make sense of tons of telemetry data. Cut through the noise to get the right answers quickly and easily. New Relic Grok leverages OpenAI's large language models (LLMs) so that any engineer can use plain language and a familiar chat interface to ask questions and get insights, without any prior observability experience. Observability is now as simple as asking New Relic Grok, "Why is my cart not working?" or "Instrument AWS."

Top 7 Most-Used Distributed System Patterns

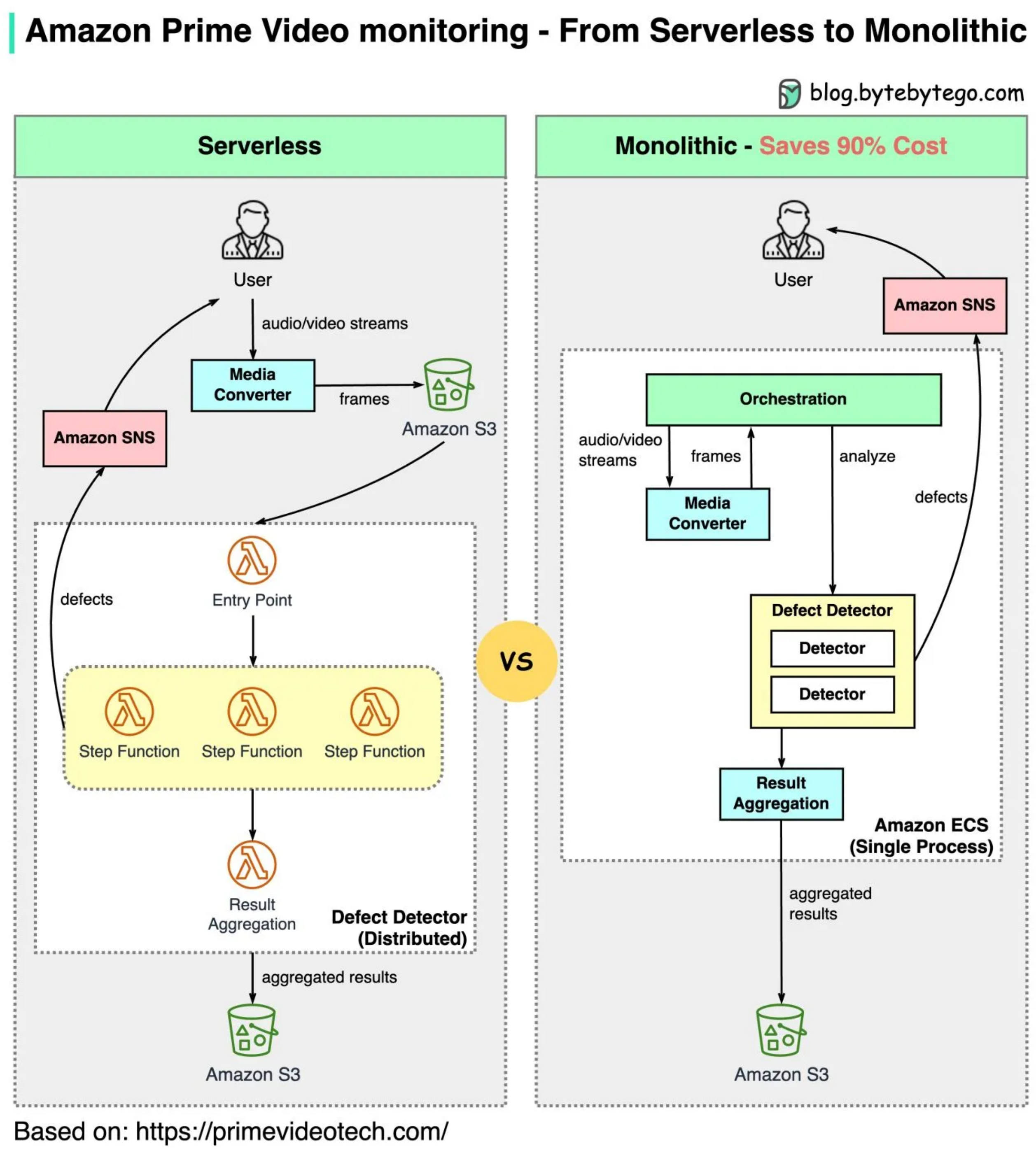
- Ambassador
- Circuit Breaker
- CQRS
- Event Sourcing
- Leader Election
- Publisher/Subscriber
- Sharding

Which additional patterns have we overlooked?

Amazon Prime Video Monitoring Service

Why did Amazon Prime Video monitoring move **from serverless to monolithic**? How can it save 90% cost?

The diagram below shows the architecture comparison before and after the migration.



What is Amazon Prime Video Monitoring Service?

Prime Video service needs to monitor the quality of thousands of live streams. The monitoring tool automatically analyzes the streams in real time and identifies quality issues like block corruption, video freeze, and sync problems. This is an important process for customer satisfaction.

There are 3 steps: media converter, defect detector, and real-time notification.

- What is the problem with the old architecture?

  The old architecture was based on Amazon Lambda, which was good for building services quickly. However, it was not cost-effective when running the architecture at a high scale. The two most expensive operations are:

  1. The orchestration workflow - AWS step functions charge users by state transitions and the orchestration performs multiple state transitions every second.

  2. Data passing between distributed components - the intermediate data is stored in Amazon S3 so that the next stage can download. The download can be costly when the volume is high.
- Monolithic architecture saves 90% cost

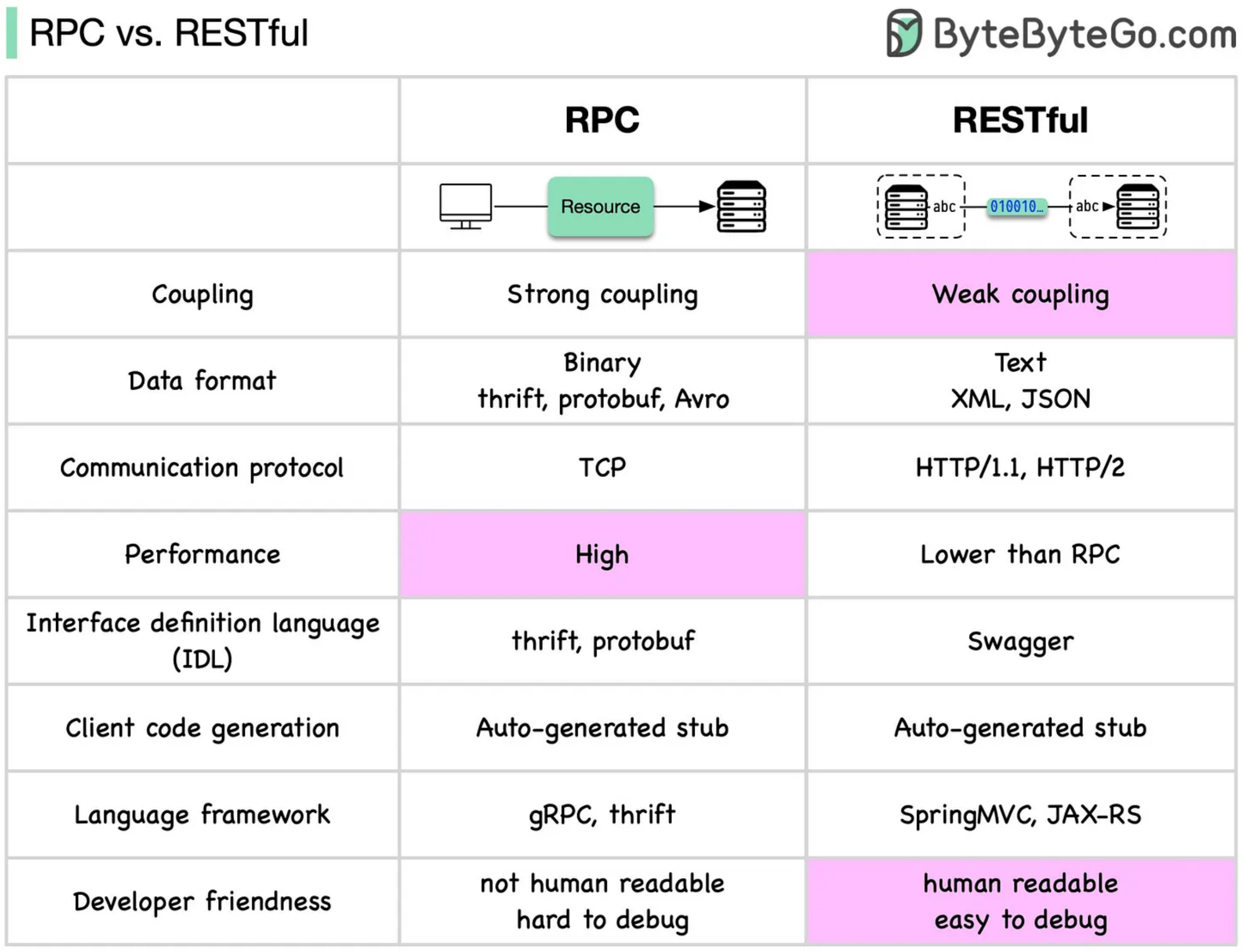
  A monolithic architecture is designed to address the cost issues. There are still 3 components, but the media converter and defect detector are deployed in the same process, saving the cost of passing data over the network. Surprisingly, this approach to deployment architecture change led to 90% cost savings!

  This is an interesting and unique case study because microservices have become a go-to and fashionable choice in the tech industry. It's good to see that we are having more discussions about evolving the architecture and having more honest discussions about its pros and cons. Decomposing components into distributed microservices comes with a cost.
- What did Amazon leaders say about this?
  Amazon CTO Werner Vogels: “Building **evolvable software systems** is a

> strategy, not a religion. And revisiting your architectures with an open mind is a must.”
>
> Ex Amazon VP Sustainability Adrian Cockcroft: “The Prime Video team had followed a path I call **Serverless First**…I don’t advocate **Serverless Only**”.

Over to you: Does microservice architecture solve an architecture problem or an organizational problem?

How to choose between RPC and RESTful?


RPC vs. RESTful
ByteByteGo.com
Resource
abc
010010...
abc

	RPC	RESTful
Coupling	Strong coupling	Weak coupling
Data format	Binary thrift, protobuf, Avro	Text XML, JSON
Communication protocol	TCP	HTTP/1.1, HTTP/2
Performance	High	Lower than RPC
Interface definition language (IDL)	thrift, protobuf	Swagger
Client code generation	Auto-generated stub	Auto-generated stub
Language framework	gRPC, thrift	SpringMVC, JAX-RS
Developer friendness	not human readable hard to debug	human readable easy to debug

Communication between different software systems can be established using either RPC (Remote Procedure Call) or RESTful (Representational State Transfer) protocols, which allow multiple systems to work together in distributed computing.

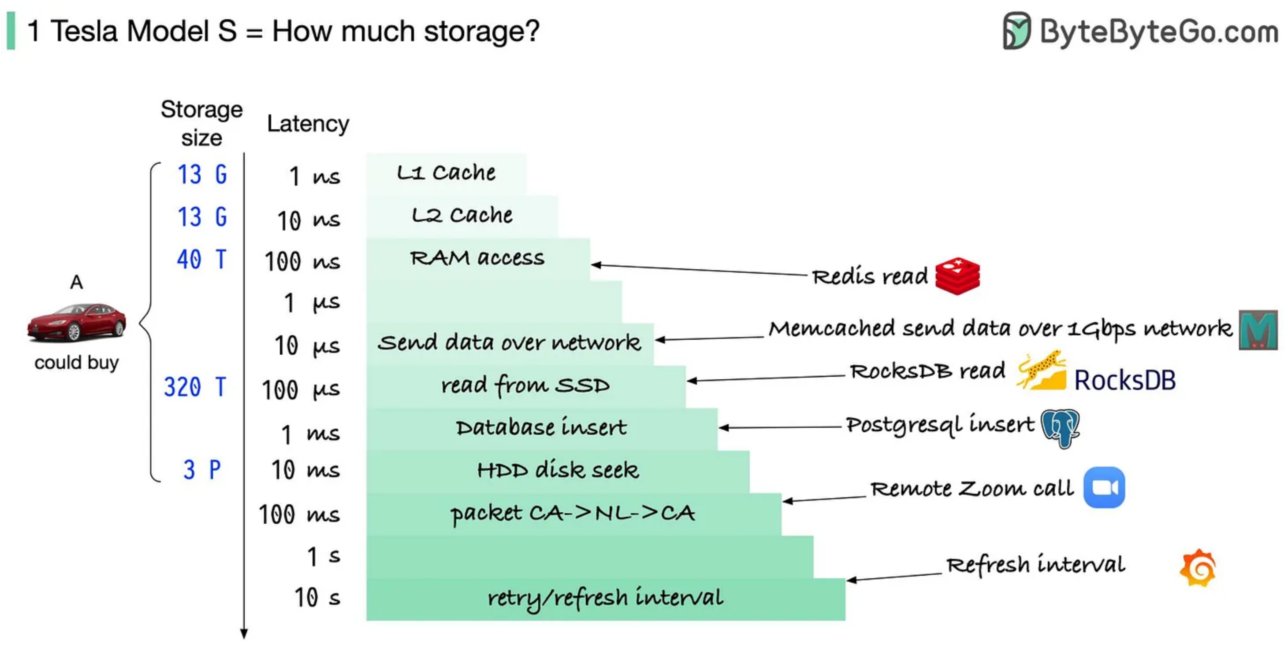
The two protocols differ mainly in their design philosophy. RPC enables calling remote procedures on a server as if they were local procedures, while RESTful applications are resource-based and interact with these resources via HTTP methods.

When choosing between RPC and RESTful, consider your application's needs. RPC might be a better fit if you require a more action-oriented approach with custom operations, while RESTful would be a better choice if you prefer a standardized, resource-based approach that utilizes HTTP methods.

Over to you: What are the best practices for versioning and ensuring backward compatibility of RPC and RESTful APIs?

How much storage could one purchase with the price of a Tesla Model S?

There's a trade-off between the price of a storage system and its access latency. Naturally, one might wonder how much storage could be obtained if one is willing to sacrifice latency.



To make this calculation more intriguing, let's use the price of a Tesla Model S as a benchmark. Here are the relevant prices:

- Tesla Model S: $87,490 per car
- L1 cache: $7 per megabyte
- L2 cache: $7 per megabyte
- RAM: $70 for 32G
- SSD: $35 for 128G
- HDD: $350 for 12T

Join the ByteByteGo Talent Collective

If you're looking for a new gig, join the collective for customized job offerings from selected companies. Public or anonymous options are available. Leave anytime.

If you're **hiring**, join the ByteByteGo Talent Collective to start getting bi-monthly drops of world-class hand-curated engineers who are open to new opportunities.

109 Likes · 3 Restacks

10 Comments

Write a comment...

Dean Schulze May 13

Where is the article about the 90% Cost Slash: From Serverless to Monolith?

LIKE (6) REPLY SHARE

2 replies

Mitch Spradlin May 14

The text of the RPC vs. RESTful section correctly identifies the two as different design methodologies, but the table is misleading. It's comparing a couple of RPC and REST frameworks, not the design approaches. For example, it’s perfectly common for RPC frameworks to use human-readable JSON and for REST APIs to use a binary serialization.

LIKE (2) REPLY SHARE

8 more comments...