How to model microservices?

Deployment modeling and optimization process for digital enterprises

wipro

In a typical digital transformation program, the number of microservices grows within a range of 500 to 1000. This huge growth in microservices affects infrastructure and capacity planning, and defining a scalable model becomes complex. In addition to this, driving optimization in IT infrastructure gets more challenging.

In traditional application capacity sizing, many vendors provide references on the capacity of application servers, with optimal performance KPIs using lab tests. For example, an application server can perform with a throughput of 1000 Transaction Per Second (TPS) with moderate processing complexity-with two quad core CPUs, two GB heap size. Many enterprises follow this method for determining the initial sizes for application servers. Some enterprises use performance-testing outputs to validate and re-size the production systems.

When it comes to microservices, especially with cloud container technologies, there are not many such standards or benchmarks readily available for capacity planning or sizing.

Many architects/designers use random methods to size the components. In many cases, microservices instances are oversized, needing large CPU and memory requirements for infrastructures. Hence, including deployment modeling and optimization process during the early test cycles of microservices is essential.

Microservices deployment modeling and its benefits

Deployment modeling is a process of right sizing and tuning Java Virtual Machine (JVM) heap sizes (like old gen space, eden space, native space), garbage collection policy, thread pool size, thread stack size, connection pool sizes etc. And identifying the optimal capacity of a single microservices component for a given environment configuration.

It is important to include the microservices component deployment modeling plan as part of the overall digital project planning. Modeling is similar to performance testing: it focuses more on identifying the maximum capacity of a single component with optimal performance and optimal memory sizing.

Some of the key benefits of microservices deployment modeling include:

Optimal performance of given containers due to right sizing of individual microservices component

Cost reduction by defining the right number of microservices instances for a given load

IT resource optimization

Early detection of potential performance improvements and bottlenecks

Improved response time of microservices APIs thus enhancing end-user experience

Improved code quality and performance of software

Reduced performance testing effort

Reduced performance incidents post production release

Microservices component deployment modeling process

The following guidelines are suggestions to aid deployment modeling of a single microservices component: use these steps as a reference and not as an industry standard.

To begin with, start with a small set of microservices components to model and baseline the performance. Later on, develop the capability to automate this exercise wherever possible as a continuous testing and modeling process. Use this method for all existing and upcoming microservices components.

- Setup a dedicated environment space for modeling single microservices components
- Identify key microservices functions and develop testing scripts (similar to performance test scripts)
- In cases where host systems are not available in the modeling environment, use service virtualization tools for simulating host responses. Induce possible delays in simulation tools to mimic production scenarios
- Define SLA for microservices
- Determine the right mix of API load as per production scenario for testing
- For baselining the microservices KPIs, test with twice the expected production volume for catering to failover, disaster recovery or cloud bursting scenarios
- Test early and test often for consistent results
- Conduct a constant load test on a single container of a long duration cycle (8 to 12 hours) to identify and fix memory leak issues
- Identify right values for JVM tunable parameters (thread size, pools, heap sizes, old gen, eden space etc)
- Establish performance baseline for single container e.g. 'single' login microservices component of 500 MB size can sustain 20 TPS of load while meeting all required SLAs
- While calculating production sizing, add appropriate contingency factors from the lab results as there could be unknown factors that could hinder the performance in a production environment e.g. If optimal capacity of 'login' microservices component in lab test is 20 TPS, then for production sizing consider the optimal capacity as 15 TPS
- Repeat this process on every major releases/code changes on the component. Compare results with previous releases for any performance degradation
- Include modeling and baselining as a standard process for new microservices components so that right-sizing and tuning of a component is done before performance testing and production deployment

Dependencies: Microservices deployment modeling requires the support of good application monitoring and log monitoring tools with detailed tracing capabilities to measure component KPIs. In most cases, such tools may already exist in an enterprise.

Value realization for deployed components

Identify opportunities to reduce the oversized containers in production. For example, let us assume that in production we have a 'login' microservice with 2 gb container heap size for handling a load of 10 TPS. Modeling exercise was not carried out on this login service previously. Now, after the modeling process in the lab, we find that 'login' microservice can handle 15 requests at a time (20 TPS) with only 500 mb container heap size. This means that the existing production container of 2 gb heap is oversized. The heap size should be reduced to 500 mb to handle the required load of 10 TPS. This will bring down memory footprint from 2 gb to 500 mb, saving 1.5 gb per container. The memory savings will be substantial if there are more of such oversized containers in production.

Microservices deployment modeling process can be well integrated with DevOps process, where this activity can be planned in parallel with any development/test cycles of a sprint.

The digital now

Although application performance baselining or modeling has been known since the early IT revolution, its applicability in modern digital enterprises is still not very common. It requires a well-defined strategy, management focus, detailed process and most importantly, patience to implement it. Once the people, processes and environments are established, the benefits are manifold.

About the author

Sandeep Tol
Senior Architect, Strategy Consulting and Architecture division, Wipro Limited

Sandeep has 19 years of IT experience in architecture, design and implementation across the breadth of Cloud, microservices, application and SOA integration areas. In his current role, he is helping large clients in reviewing microservices architectures and building shared microservices models.

Wipro Limited
Doddakannelli, Sarjapur Road,
Bangalore-560 035, India

Tel: +91 (80) 2844 0011
Fax: +91 (80) 2844 0256
wipro.com

Wipro Limited (NYSE: WIT, BSE: 507685, NSE: WIPRO) is a leading global information technology, consulting and business process services company. We harness the power of cognitive computing, hyper-automation, robotics, cloud, analytics and emerging technologies to help our clients adapt to the digital world and make them successful. A company recognized globally for its comprehensive portfolio of services, strong commitment to sustainability and good corporate citizenship, we have over 160,000 dedicated employees serving clients across six continents. Together, we discover ideas and connect the dots to build a better and a bold new future.

For more information, please write to us at **info@wipro.com**

IND/BRD/JAN 2018-DEC 2017