Day 1 - What is Performance Testing

Apa itu Performance Testing

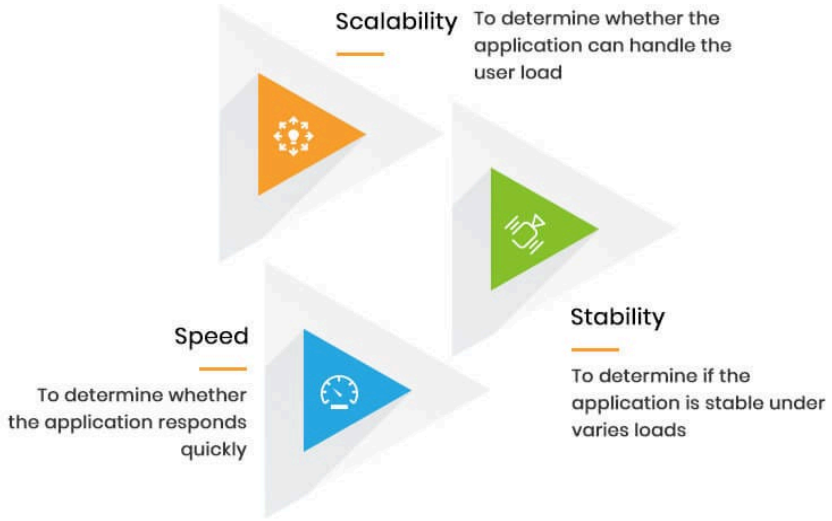
Performance testing, sesuai dengan namanya, adalah pengujian non-functional yang dilakukan untuk mengetahui behavior atau performa dari sebuah sistem saat dihadapkan pada situasi yang bervariasi sehingga dapat menghindari kegagalan sistem seperti contohnya system down.

“Performance testing is critical to customer satisfaction; if your application’s performance doesn’t meet the expectations of your customers, they will move on to your competitor.” — Anonymous

Pada pengujian ini, seorang software tester / QA tidak dituntut untuk mencari error melainkan mencari hambatan-hambatan yang dapat membuat sebuah aplikasi loading terlalu lama atau mungkin gagal diakses. Aplikasi yang loading-nya lama tentu akan memberikan penilaian yang buruk pada user.

Dalam performance testing ada beberapa faktor yang menjadi fokus pada pengujian ini, diantaranya :

- **Speed** : melihat kecepatan sistem dalam memberikan respon dari setiap request
- **Scalability** : menentukan berapa jumlah load/user/thread maksimum yang dapat ditangani sistem
- **Stability** : menganalisis kondisi sistem saat diberikan beban yang bervariasi, apakah stabil atau tidak



https://miro.medium.com/v2/resize:fit:800/1*JGvJPc7f9oob3VcKv82_lw.jpeg

Tujuan Performance Testing

Tujuan dari Performance Testing adalah untuk mengevaluasi dan mengukur kinerja suatu aplikasi atau sistem komputer dalam berbagai situasi dan kondisi beban (load) kerja. Beberapa tujuan utama dari Performance Testing yaitu:

1. **Mengidentifikasi Batasan Kapasitas**

   Performance Testing membantu mengungkap batasan kapasitas suatu aplikasi atau sistem. Dengan menentukan sejauh mana sistem dapat menangani jumlah user atau volume data tertentu, pengembang dapat mengidentifikasi titik - titik lemah yang perlu ditingkatkan.

2. **Memvalidasi Kecepatan Respon**

   Tujuan lainnya adalah memastikan bahwa waktu respons aplikasi tetap dalam batas yang dapat diterima.

   User pasti mengharapkan aplikasi untuk merespons dengan cepat, dan Performance Testing membantu memastikan bahwa aplikasi tidak mengalami penurunan kinerja yang signifikan saat digunakan oleh banyak user secara bersamaan.

   Berikut ini adalah contoh gambaran standar waktu respons yang berlaku secara umum:

   - **Respon Instan:** Kurang dari 1 detik – User merasakan interaksi langsung.
   - **Respon Cepat:** 1-2 detik – User merasakan sedikit penundaan, tetapi masih dalam batas toleransi.
   - **Respon Sedang:** 2-5 detik – User mulai merasakan penundaan, yang dapat mengurangi kenyamanan.
   - **Respon Lambat:** Lebih dari 5 detik – User mungkin merasa frustasi dan tidak puas.

   Standar ini membantu menetapkan ekspektasi kinerja dan memberikan panduan bagi pengembang untuk meningkatkan user experience.

3. **Menilai Stabilitas**

   Performance Testing membantu menilai stabilitas sistem di bawah tekanan dan beban kerja tinggi. Ini memungkinkan pengembang untuk mengidentifikasi potensi crash pada sistem atau masalah yang mungkin timbul selama operasi normal atau dalam situasi kritis.

4. **Mengukur Skalabilitas**

   Performance Testing juga bertujuan untuk mengukur skalabilitas suatu sistem, yaitu kemampuannya untuk menangani peningkatan jumlah user atau beban kerja tanpa mengorbankan kinerja atau waktu respons.

5. **Mendukung Pengambilan Keputusan**

   Hasil dari Performance Testing memberikan pemahaman yang lebih baik tentang kinerja aplikasi atau sistem, membantu tim pengembang dan manajemen dalam pengambilan keputusan terkait peningkatan kinerja, pengoptimalan kode, atau penyesuaian infrastruktur.

Pentingnya Performance Testing

Performance Testing sangat penting dalam pengembangan perangkat lunak karena berkontribusi secara signifikan untuk memastikan bahwa suatu aplikasi atau sistem dapat berkinerja optimal di bawah berbagai kondisi. Berikut adalah beberapa alasan mengapa Performance Testing diperlukan:

1. **Memastikan Pengalaman User yang Baik**

   Performance Testing membantu memastikan bahwa user dapat mengakses dan menggunakan aplikasi dengan lancar tanpa mengalami keterlambatan atau penurunan kualitas layanan.

   Waktu respons yang cepat dan kinerja yang stabil adalah kunci untuk memberikan pengalaman user yang baik.

2. **Mendeteksi Potensi Masalah Sejak Dini**

   Dengan melakukan performance testing, tim pengembang dapat mendeteksi potensi masalah atau bottleneck sejak dini. Hal ini memungkinkan mereka untuk mengidentifikasi dan mengatasi masalah sebelum aplikasi atau sistem diluncurkan ke end user.

3. **Menemukan Batasan Kapasitas**

   Performance Testing membantu mengungkap batasan kapasitas suatu aplikasi atau sistem. Ini melibatkan test untuk menentukan sejauh mana sistem dapat menangani jumlah user atau volume data tertentu tanpa mengalami penurunan kinerja yang signifikan.

4. **Mengoptimalkan Kinerja Aplikasi**

   Hasil dari Performance Testing memberikan wawasan tentang area yang perlu dioptimalkan. Ini dapat mencakup perbaikan pada kode, penyesuaian infrastruktur, atau pengoptimalan konfigurasi untuk meningkatkan kinerja keseluruhan aplikasi.

5. **Menghadapi Beban Kerja yang Tinggi**

   Performance Testing membantu mengidentifikasi sejauh mana suatu aplikasi atau sistem dapat menangani beban kerja yang tinggi. Ini sangat penting dalam situasi di mana jumlah user atau volume transaksi dapat meningkat secara tiba-tiba, seperti pada saat promosi atau kejadian khusus.

6. **Mengukur Ketersediaan dan Ketahanan**

   Performance Testing juga menilai ketersediaan dan ketahanan suatu sistem di bawah tekanan. Dengan mengidentifikasi area-area yang rentan terhadap kegagalan, tim pengembang dapat mengambil langkah-langkah untuk meningkatkan keandalan aplikasi.

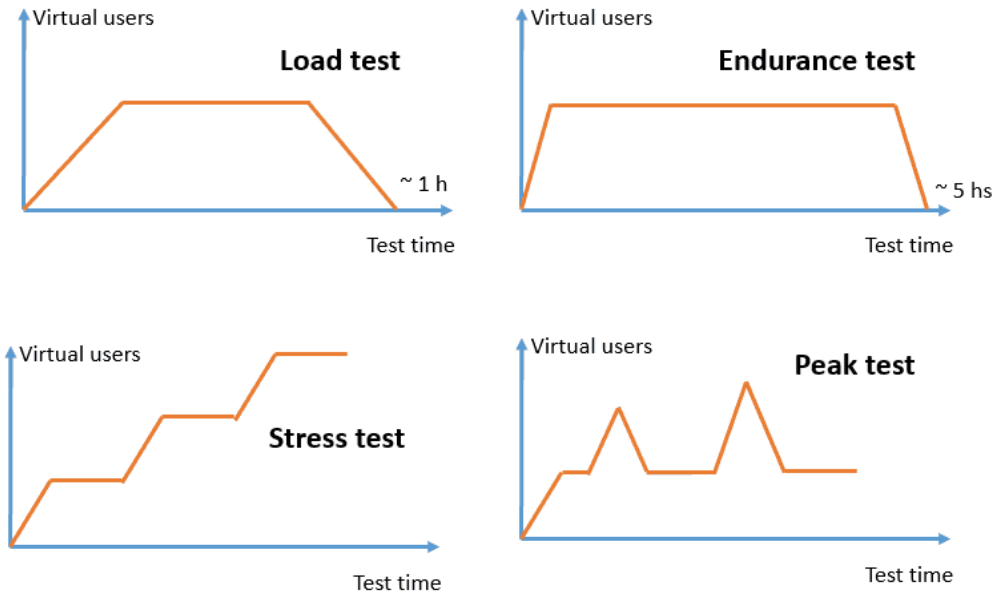
7. **Mendukung Pengambilan Keputusan**

   Hasil dari Performance Testing memberikan data yang diperlukan untuk pengambilan keputusan yang informasional. Manajemen dapat menggunakan informasi ini untuk menentukan apakah perlu dilakukan investasi tambahan dalam infrastruktur atau pengoptimalan aplikasi.

8. **Mengurangi Risiko Downtime**

   Dengan mengidentifikasi dan mengatasi potensi masalah sejak dini, Performance Testing membantu mengurangi risiko downtime yang dapat merugikan bisnis. Ini meningkatkan ketersediaan aplikasi dan mencegah hilangnya peluang atau kehilangan kepercayaan user.

Jenis Performance Testing



Some Types of Performance Tests
https://abstracta.us/wp-content/uploads/2015/03/why-perf-testing-is-nec-chart.png

Ada berbagai macam performance testing. Hal ini dibedakan berdasarkan goal dan cara melakukan testnya.

Disclaimer: setiap organisasi mungkin memiliki istilah yang berbeda. Namun, pada dasarnya definisinya sama.

Berikut adalah beberapa jenis Performance Testing yang umum dilakukan:

1. **Load Testing (Uji Beban)**

   Load Testing melibatkan pemberian beban pada suatu sistem untuk mengukur responnya. Tujuan utama adalah untuk menilai sejauh mana sistem dapat menangani jumlah user atau beban kerja tertentu tanpa mengalami penurunan kinerja atau kegagalan. Pengujian ini diperlukan saat adanya sebuah event musiman yang menyebabkan lalu lintas (traffic) sistem bertambah seiring berjalannya waktu.

   Contohnya, pada sebuah aplikasi penjualan tiket yang traffic-nya perlahan akan bertambah saat sebuah event mulai mendekati. Misalnya acara tahun baru yang terjadi di tanggal 1 Januari, mungkin jauh-jauh hari seiring mendekatnya tahun baru, traffic dari aplikasi penjualan tiket juga akan bertambah sehingga pengujian ini diperlukan untuk mengetahui ketahanan sistem dengan menaikkan load perlahan-lahan pada setiap skenario-nya, kemudian menganalisis penggunaan resource-nya apakah ada perbedaan yang signifikan atau tidak.

2. **Stress Testing (Uji Stres)**

   Stress Testing menguji batasan suatu sistem dengan memberikan beban di atas kapasitas normalnya. Hal ini dilakukan untuk mengidentifikasi titik kegagalan atau batasan kapasitas yang mungkin menyebabkan sistem tidak berfungsi dengan baik atau bahkan mengalami crash. Salah satu alasan utama dilakukannya pengujian ini adalah untuk menghindari terjadinya kegagalan sistem (system down).

   Contoh kasus yang paling umum adalah saat sebuah aplikasi e-commerce mengadakan promo besar-besaran, sehingga banyak user yang mengakses aplikasi tersebut di jam tertentu (misalnya ada waktu tertentu untuk menggunakan promo) pada waktu yang bersamaan. Setelah dilakukannya pengujian ini, diharapkan tim pengembang dapat membangun sebuah sistem yang lebih kebal terhadap kasus traffic yang dibanjiri oleh banyak user.

3. **Endurance Testing (Uji Daya Tahan)**

   Endurance Testing merupakan bagian dari teknik pengujian performance yang dijalankan untuk menganalisis behavior atau performa sistem dengan load yang normal namun dalam jangka waktu yang panjang.

   Pengujian ini dilakukan untuk menghindari terjadinya kebocoran memori (memory leak). Di beberapa instansi, endurance testing tidak menjadi bagian dari tanggung jawab seorang Software Tester/QA. Pengujian ini biasanya dilakukan oleh seorang performance engineer, atau role yang lebih ahli di bidang ini.

4. **Scalability Testing (Uji Skalabilitas)**

   Scalability Testing mengukur kemampuan suatu sistem untuk berkembang dan menangani peningkatan jumlah user atau beban kerja. Ini membantu dalam menentukan sejauh mana sistem dapat dikembangkan tanpa mengorbankan kinerja.

5. **Volume Testing (Uji Volume)**

   Volume Testing menguji kinerja suatu aplikasi atau sistem saat dihadapkan dengan volume data yang besar. Tujuannya adalah untuk memastikan bahwa sistem dapat mengelola dan memproses jumlah data yang besar tanpa menurunkan kinerja atau mengalami kegagalan.

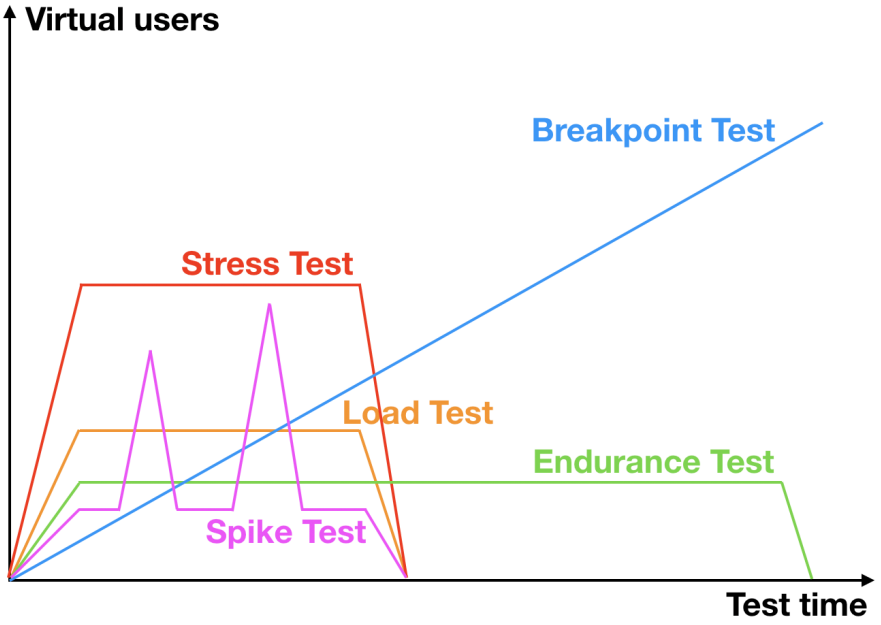
   Tes ini dilakukan untuk melihat bagaimana behaviour sistem ketika memiliki data yang sangat besar. Biasanya tes dilakukan bertahap dimulai dari jumlah data yang masih sedikit di database, kemudian akan kita tambah terus menerus datanya sampai jumlahnya sangat besar. Tujuannya tidak lain untuk kesiapan sistem secara

long term. Dimana semakin lama sistem digunakan, biasanya akan semakin besar data yang disimpan.

6. **Spike Testing**

   Seperti stress test, namun kenaikan langsung menuju melebihi puncak dilakukan dalam waktu singkat. Jika stress test memberikan waktu untuk scale-out, disini tidak.

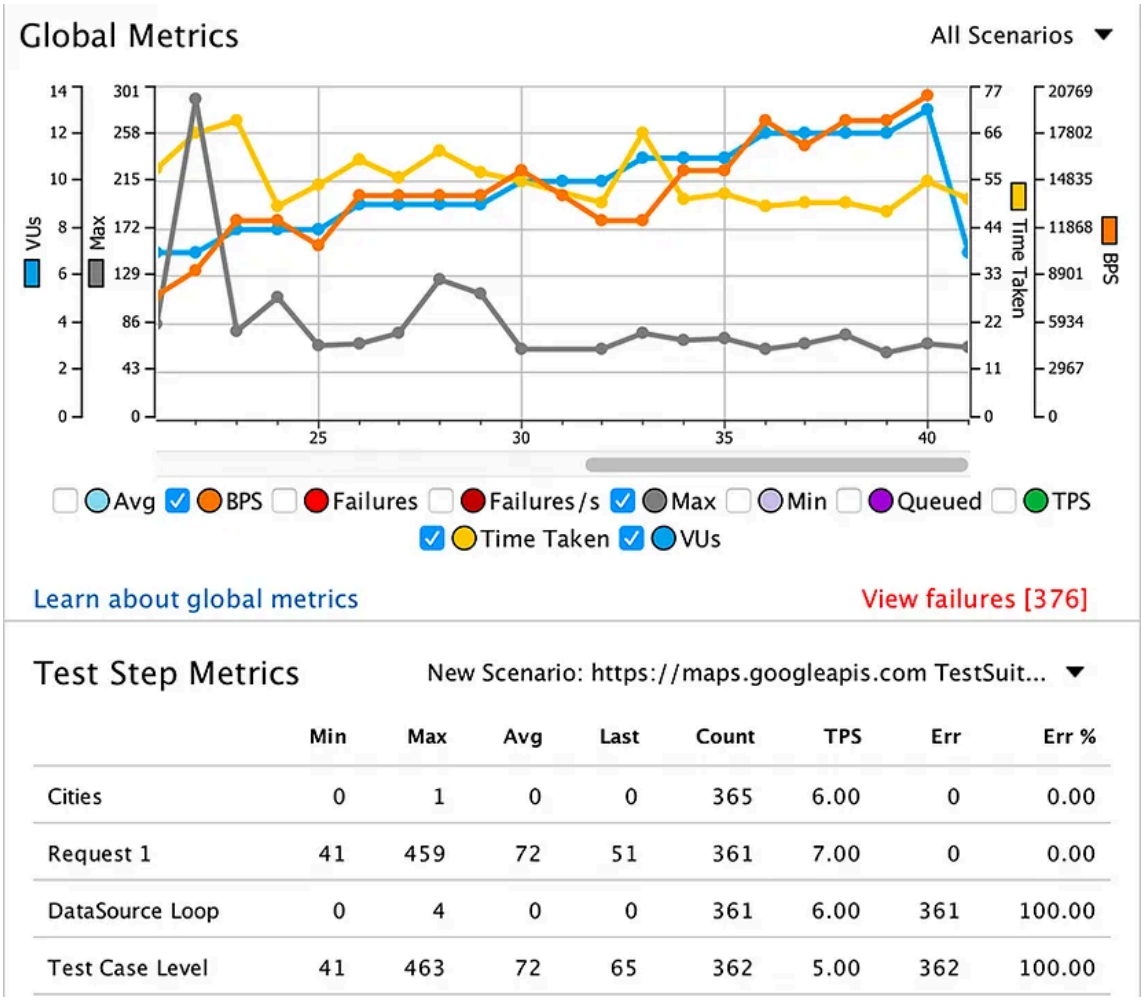
   Salah satu contoh adalah pada saat flash sale di suatu e-commerce, yang dimana banyaknya kunjungan user yang langsung memuncak saat mengakses halaman flash sale.



Comparison of Some Performance Tests Types

https://abstracta.us/wp-content/uploads/2015/03/why-perf-testing-is-nec-chart.png

Matriks Performance Testing



	Min	Max	Avg	Last	Count	TPS	Err	Err %
Cities	0	1	0	0	365	6.00	0	0.00
Request 1	41	459	72	51	361	7.00	0	0.00
DataSource Loop	0	4	0	0	361	6.00	361	100.00
Test Case Level	41	463	72	65	362	5.00	362	100.00

https://smartbear.com/blog/2020/measuring-api-performance-with-loadui-pro/?lang=de-de

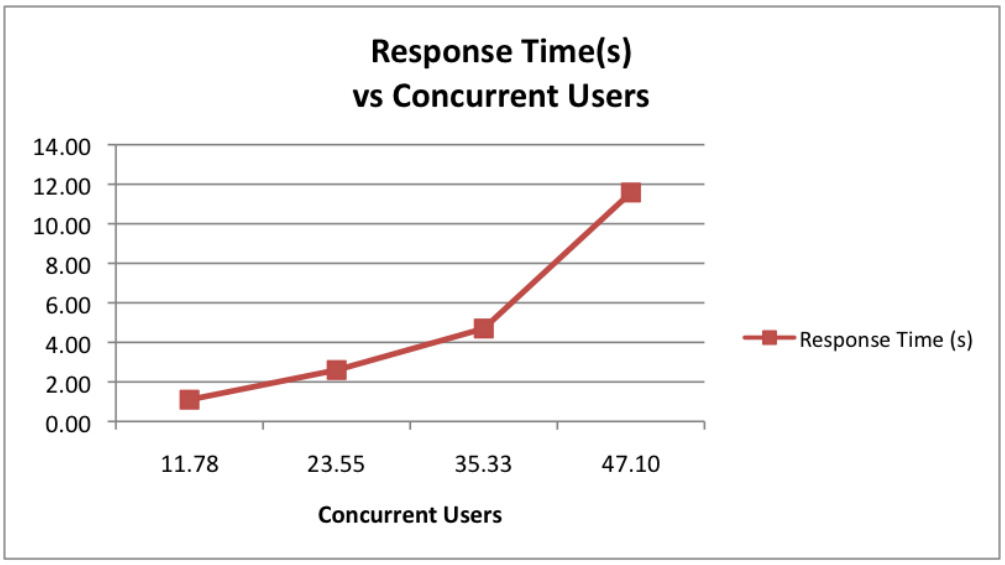
Ada beberapa metrik performance testing yang penting untuk dipahami dengan benar oleh team developer, QA, atau team penguji untuk dapat menarik kesimpulan dengan benar dan tepat dari data yang di dapatkan.

Besaran atau jenis data apa saja yang harus kita hitung selama performance test dilakukan?

Berdasarkan data yang diberikan dari tools performance test dan dari beberapa literatur, umumnya performance test akan menghitung besaran-besaran seperti berikut.

1. **Response Time**

   Waktu yang berlalu dari saat request masuk ke server sampai hingga byte terakhir diterima kembali dari server disebut response time. Besaran atau metrik ini diukur dalam kilobyte per detik (KB/detik). Respon time akan meningkat seiring dengan peningkatan beban user.



Response Time(s) vs Concurrent Users

http://s3.amazonaws.com/danvalencia_my_site/response_time_concurrency.png

2. **Jumlah Request Per Satuan Waktu (RPS/ RPM)**

   Aplikasi client memberikan request HTTP dan mengirimkannya ke server. Software server akan merespon dan mengirimkannya kembali ke client. Jumlah total request yang konsisten per detik disebut sebagai RPS (request per second) semakin besar nilainya, menunjukan semakin bagus kualitas dari software yang sedang di test. Terkadang Request ini dihitung dalam satuan menit di sebut RPM (request per menit). Adapun, response data yang diberikan bisa untuk semua jenis response baik JSON, xml, atau yang lainnya.

3. **User Transaction**

   User transaction merupakan urutan aksi atau tindakan dari user melalui antarmuka (UI) software. Besaran ini. didapatkan dengan membandingkan waktu transaksi aktual dengan waktu yang diharapkan. Kita dapat menyimpulkan seberapa sukses sistem telah lulus load testing dengan mengetahui user transaction ini.

4. **Error Rate**

   Besaran ini digunakan untuk mengetahui rasio atau perbandingan antara respon yang benar dan respon yang gagal selama periode tertentu. Hasilnya dihitung dalam bentuk persentase (%).

   Umumnya, kesalahan terjadi ketika load yang diterima mulai melebihi kapasitas. Selain itu bisa saja terjadi kesalahan ketika beban atau concurrency meningkat. Semakin kecil error rate semakin bagus sistem.

5. **Virtual User Per Unit Waktu**

   Jumlah virtual data dihitung dengan rumus hasil perkalian durasi waktu user (dalam detik) dengan jumlah request per detik. Untuk memahaminya bisa melihat dari contoh berikut.

   Beban puncak pada aplikasi adalah **10 user per jam** dan setiap user rata-rata menghabiskan **10 menit** di situs web dan menelusuri **10 halaman web**. Ini berarti skenario user yang diterapkan akan berisi 10 request halaman dan setiap user virtual akan berjalan selama 10 menit.

   Rumus di atas dalam kasus seperti itu akan dimodifikasi menjadi:

   *Jumlah virtual user = Jumlah user per jam * Jumlah request per user * 3600 / [Panjang skenario user (dalam detik)]*

   Di sini:

   - Jumlah user per jam = 10
   - Jumlah request per user = 10 (jumlah halaman web yang diakses)
   - Durasi skenario user = 10 menit * 60 = 600 detik

   Oleh karena itu menggunakan rumus di atas:

   *Jumlah virtual user = 10 * 10 * 3600 / 600 = 600*

   Besaran ini juga membantu untuk mengetahui apakah kinerja dari software memenuhi persyaratan yang diminta. Ini membantu tim QA untuk memperkirakan beban rata-rata serta perilaku software dalam kondisi beban yang berbeda.

6. **Wait time (Latency)**

   Dikenal juga sebagai latensi rata-rata. Wait time memberitahu berapa lama waktu berlalu dari mulai request di kirim ke server sampai byte pertama data masuk ke client. Di hitung sama dalam KB/detik. Hati-hati jangan salah pengertiannya dengan response time.

   Contoh Ilustrasi

   - **Web Page Load:**
     - **Latency:** Waktu yang diperlukan untuk browser mengirim HTTP request ke server dan menerima respons awal dari server. Dipengaruhi oleh faktor-faktor seperti jarak fisik antara klien dan server, kecepatan jaringan, dan kualitas koneksi jaringan.

- **Response Time:** Waktu total dari saat user mengklik link hingga halaman web sepenuhnya dimuat dan dapat digunakan, termasuk waktu pemrosesan server dan rendering halaman oleh browser. Dipengaruhi oleh beberapa faktor, termasuk latency, kecepatan pemrosesan server, dan efisiensi kode aplikasi.

7. Rata-Rata Load Time

Merupakan rata-rata waktu yang diperlukan untuk mengirimkan request. Rata-rata load time ini merupakan salah satu parameter bagi end user untuk mengetahui kualitas dari software yang digunakan.

Misalnya, jika halaman web membutuhkan waktu lebih dari tiga detik untuk dimuat, kemungkinan besar seseorang akan meninggalkannya. Untuk memastikan hal ini tidak terjadi, tim QA harus mengukur waktu muat rata-rata dan menyarankan area untuk pengoptimalan jika halaman dimuat terlalu lambat kepada team developer.

8. Concurrent Users

Dikenal sebagai ukuran beban, besaran ini menunjukan penggunaan aktif di bagian manapun. Ini adalah salah satu metrik yang paling banyak digunakan untuk mempelajari perilaku software di bawah sejumlah virtual user.

Nilainya akan mempengaruhi request per satuan waktu. Pada kasus concurrent user ini QA team tidak men generate request yang konsisten. Ada waktu jeda sehingga semua request masuk ke server tidak secara simultan tetapi datang secara berurutan dengan jeda waktu yang singkat.

9. Throughput

Throughput menunjukkan bandwidth yang digunakan selama pengujian. Dengan kata lain, ini menunjukkan jumlah maksimum data yang mengalir melalui koneksi jaringan tertentu dalam waktu tertentu.

Kita mengukur throughput dalam KB/detik. Metrik ini sering kali bergantung pada jumlah user secara bersamaan. Umumnya throughput dinyatakan dalam rps atau rpm.

10. Penggunaan CPU

Seperti yang dapat kita tebak dengan mudah berdasarkan namanya, metrik ini menunjukkan berapa banyak waktu yang digunakan CPU untuk memproses request tertentu. Bisa juga dinyatakan dengan seberapa banyak core CPU yang digunakan untuk menyelesaikan suatu task tertentu.

Harusnya dengan bertambahnya user yang menggunakan pada batas-batas tertentu response time nya masih sama tetapi penggunaan cpu nya bertambah. (Sistem berjalan dengan semestinya). Jika dengan bertambahnya user response time melambat tetapi penggunaan cpu nya tetap berarti terdapat cara pengkodean yang tidak optimal, terdapat blocking, dan request yang mengantri.

11. **Penggunaan Memori**

    Demikian pula, pemanfaatan memori menunjukkan berapa banyak sumber daya yang diperlukan untuk memproses request, tetapi dalam hal memori pada server tempat software di test bukan banyaknya memori pada client yang digunakan untuk menjalankan tes nya.

    Sama seperti pada CPU harusnya semakin bertambah user penggunaan memori akan bertambah. Akan tetapi, jika dengan bertambahnya user response time melambat tetapi penggunaan memorinya tetap berarti ada cara pengkodean yang tidak optimal, terdapat blocking, dan request yang mengantri.

Permasalahan Umum Performance Testing

Sebagian besar masalah kinerja berkisar pada ***kecepatan, waktu response, waktu load, skalabilitas, dan stabilitas yang buruk***. Kecepatan sering kali menjadi salah satu atribut terpenting dari sebuah aplikasi. Aplikasi yang berjalan lambat akan kehilangan calon pengguna. Pengujian kinerja dilakukan untuk memastikan aplikasi berjalan cukup cepat untuk menjaga perhatian dan minat pengguna.

Lihatlah daftar masalah kinerja umum berikut dan perhatikan bagaimana kecepatan merupakan faktor umum di banyak masalah tersebut:

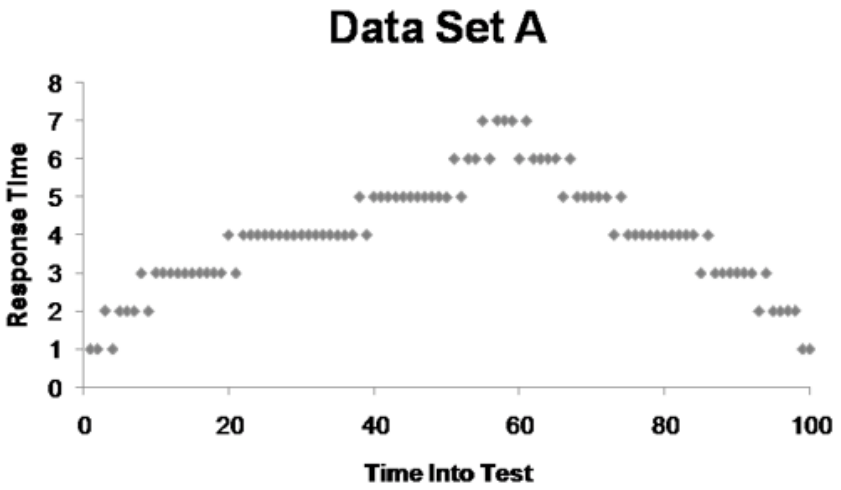
1. Load time yang lama.
2. Response time yang jelek
3. Skalabilitas yang jelek
4. Bottleneck atau blocking proses
5. Error rate yang tinggi

Analisis Data Sebagai Grafik

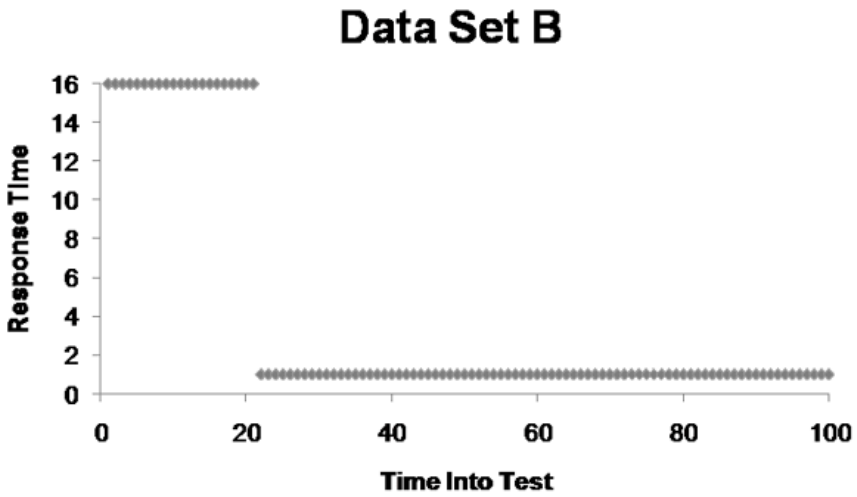
Umumnya data hasil pengujian yang didapatkan dari beberapa tool performance test seperti berikut.

	Sample Size	Minimum	Maximum	Average	Median	Normal	Mode	95th Percentile	Standard Deviation
Data Set A	100	1	7	4	4	4	4	6	1.5
Data Set B	100	1	16	4	1	3	1	16	6.0
Data Set C	100	0	8	4	4	1	3	8	2.6

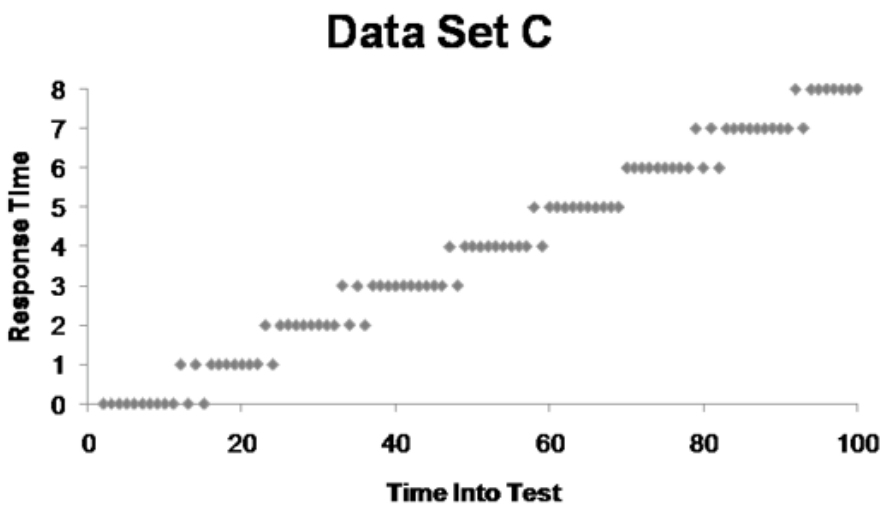
Kalau ditanya mana sistem yang memiliki kinerja baik? Kalau di lihat dari tabel di atas masih cukup sulit tetapi beda jika disajikan dalam bentuk grafik.



Dari gambar data set A dapat dilihat terdapat puncak tetapi sistem kemudian pulih kembali.



Di Set B, tampaknya dimulai dengan waktu respons yang sangat buruk dan mungkin 20 detik setelah pengujian, sistem down, dan mulai merespons halaman kesalahan.



Akhirnya, di Set C, jelas bahwa seiring berjalannya waktu, kinerja sistem terus menurun. Dari sini ternyata dengan menyajikan data dalam bentuk grafik lebih mudah untuk di fahami daripada bentuk tabel saja.

Memahami Metrik Utama Kinerja

Kita perlu memahami pengertian dari beberapa metrik kinerja yang umum agar kita mendapatkan pencerahan dari data-data yang ada.

1. **Rata-Rata**

   Untuk menghitung rata-rata, dengan menjumlahkan semua nilai sampel dan kemudian dibagi dengan jumlah sampel. Katakanlah kita melakukan test dan rata-rata yang kita hasilkan adalah 3 detik.

   Terdapat masalah dengan hasil pada nilai nominal, data memberi kita perasaan yang tidak enak bahwa semua waktu respons adalah sekitar tiga detik, terdapat beberapa lebih sedikit dan beberapa kurang sedikit, tetapi mungkin tidak demikian.

   Bayangkan kita memiliki tiga sampel, dua yang pertama dengan waktu respons satu detik, yang ketiga dengan waktu respons tujuh:

   1 + 1 + 7 = 9

   9/3 = 3

   Ini adalah contoh yang sangat sederhana yang menunjukkan bahwa tiga nilai yang sangat berbeda dapat menghasilkan rata-rata tiga, namun nilai individual mungkin tidak mendekati 3.

Hati-hati dalam menggunakan rata-rata. Karena bisa saja kita kehilangan sistem kita seperti cerita lucu berikut.

Jika kita memasukan tangan kiri kita ke dalam Es yang bersuhu -100 derajat C dan memasukan tangan kanan kita ke lahar yang bersuhu 100 derajat C. Secara rata-rata suhu-nya aman bagi tangan kita. Tapi kenyataannya, kita kehilangan dua tangan kita.

Itulah alasan mengapa tidak disarankan menggunakan SLA berdasarkan rata-rata. Disarankan menggunakan persentil data misal p(90) atau p (95) atau min dan max.

2. Percentiles

Persentil adalah metrik pengujian kinerja yang sangat berguna yang memberikan ukuran di mana persentase sampel ditemukan. Misalnya, persentil ke-90 (disingkat p90) menunjukkan bahwa 90% sampel berada di bawah nilai tersebut dan nilai lainnya (yaitu, 10% lainnya) berada di atasnya. Misal p(90) adalaj 300ms. Berarti 90% response time berada dibawah 300 ms dan 10% berada di atas 300 ms.

Ini praktis untuk menganalisis lebih dari satu nilai persentil (JMeter dan Gatling melakukannya dalam laporan mereka), menunjukkan berapa banyak sampel p90, p95, dan p99. Jika ini dilengkapi dengan minimum, maksimum dan rata-rata, maka dimungkinkan untuk memiliki tampilan data yang jauh lebih lengkap dan dengan demikian memahami bagaimana sistem berperilaku.

Beberapa persentil memiliki nama tertentu, seperti p100 yang merupakan nilai maksimum (100% data berada di bawah nilai ini), dan p50, yang merupakan median (setengah dari data berada di bawah dan setengah di atas).

3. Standar Deviasi

Standar deviasi adalah ukuran dispersi sehubungan dengan rata-rata, seberapa besar nilai-nilai bervariasi sehubungan dengan rata-rata mereka atau seberapa jauh mereka.

Jika nilai simpangan bakunya kecil, hal ini menunjukkan bahwa semua nilai sampel mendekati rata-rata, tetapi jika besar maka jaraknya berjauhan dan memiliki jangkauan yang lebih besar.

Jika semua nilainya sama, maka simpangan bakunya adalah 0. Jika ada nilai yang sangat tersebar, misalnya, pertimbangkan 9 sampel dengan nilai dari 1 hingga 9 (1, 2, 3, 4, 5, 6, 7, 8, 9), simpangan bakunya adalah ~ 2,6.

Meskipun nilai rata-rata sebagai metrik dapat sangat ditingkatkan dengan juga memasukkan standar deviasi, terlebih lagi, nilai persentil yang berguna juga.

Jadi, sebelum kita menganalisis hasil pengujian kinerja berikutnya, **pastikan untuk mengingat pertimbangan utama berikut:**

- Jangan pernah menganggap rata-rata sebagai *nilai* yang harus diperhatikan, karena bisa menipu, karena sering menyembunyikan informasi penting.
- Pertimbangkan standar deviasi untuk mengetahui seberapa berguna rata-rata, semakin tinggi standar deviasi, semakin tidak berarti.
- Amati nilai persentil dan tentukan kriteria penerimaan berdasarkan itu, dengan mengingat bahwa jika kita memilih persentil ke-90, pada dasarnya kita mengatakan, *"Kita tidak peduli jika 10% pengguna kita mengalami waktu respons yang buruk"* .

Performance Testing Tools

Some Known Tools

Ada berbagai tools yang dapat digunakan untuk melakukan Performance Testing, membantu tim pengembang dan tester dalam mengevaluasi kinerja aplikasi atau sistem. Berikut adalah beberapa tools Performance Testing yang umum digunakan:

1. **Apache JMeter**

   Open source tools yang dapat digunakan untuk melakukan load test dan mengukur kinerja aplikasi. Mendukung berbagai protokol seperti HTTP, HTTPS, FTP, JDBC, dan lainnya.

2. **LoadRunner**

   Produk dari Micro Focus yang menyediakan berbagai fitur untuk performance dan load testing. Dapat mensimulasikan beban kerja dari ribuan user untuk mengukur respons sistem.

3. **Gatling**

   Open source tools yang dirancang untuk performance dan load testing. Menyediakan skenario pengujian yang dapat disesuaikan dan berbasis skrip Scala.

4. **BlazeMeter**

   Platform load test yang memungkinkan untuk membuat, menjalankan, dan menganalisis load testing dalam skala besar. Mendukung integrasi dengan tools pengembangan perangkat lunak seperti Jenkins.

5. **Locust**

   Open source tools untuk load testing yang ditulis dalam bahasa pemrograman Python. Memungkinkan user untuk menulis skenario load testing sebagai kode Python.

6. **K6**

   Open source tools yang fokus pada load testing untuk aplikasi web dan API. Memiliki UI baris perintah yang sederhana dan dapat diintegrasikan dengan CI/CD tools.

Masing-masing tools ini memiliki tujuan yang berbeda pula. Misalnya untuk melakukan load test, dapat menggunakan JMeter, WebLoad. Sedangkan untuk melakukan stress test, dapat menggunakan LoadView, JMeter, Silk Performer.

Dalam bootcamp kali ini, kita akan menggunakan Apache JMeter karena tool ini dapat menjalankan load & stress testing, dan sudah memenuhi kebutuhan seorang tester untuk melakukan performance testing.

Apache JMeter juga sangat populer karena bersifat open source, cukup mudah digunakan, tersedia banyak jenis listener sehingga user bebas memilih listener yang dibutuhkan, dan support beberapa protokol seperti HTTP, HTTPS, XML, SOAP, dan sebagainya.

Tugas Day 1

1. Selain jenis-jenis performance testing diatas, apalagi jenis Performance Testing lain yang Kamu ketahui ?
2. Berikan minimal 1 contoh lainnya untuk performance testing sesuai dengan jenisnya masing-masing (minimal 2 jenis test)!