Teknikker for testing og forbedering av ytelse inkludert mellomlager-mekanismer for web.

Fra “Designing Data-Intensive Web Applications”

Kapittel 10. Arkitektur Design

Presentasjon av Hallvar Helleseth (hallvar@ii.uib.no)

18. Oktober 2004

INF329 – Utalgte emner i programutviklingsteknologi: Web-teknologi

Arbeidsmengde

Ytelses-spesifikasjon:

- Sideforespørsler per. tid (gjennomsnittlig og maks)
- Antall samtidige brukere
- Ventetid på svar (TTFB og TTLB)

Husk:

- Foreskjellige sider har forskjellig kompleksitet. Ta med sannsynligheten for at en side vil bli forespurt.

Ytelses måling og testing

- Trenger en implementasjon først!

  Løsning:

  - Kan måle med en prototype av applikasjonen.
  - Teste helt i slutt fasen.
- Simulere virkelige brukere og brukermønster.
  - Krever verktøy:
    - Microsoft Web Application Stess (WAS)
    - ap fra Apache

Lokalisere og fjerne flaskehalser

- Flaskehals: Når en komponent ikke svarer andre komponenter raskt nok.
- Arbeidsmengden bør være likt fordelt over ressursene.
- Flaskehalsen kan ligge i:
  - Webserver
  - Skript programmene
  - Malene for siden
  - Databasen
  - Nettverket mellom komponentene

Oppskrift:

- Øk belastningen på systemet. F.eks øk antall brukere.
- Mål “throughput”: arbeidsmengden applikasjonen får gjennomført pr tidsenhet. F.eks totalt antall sider pr. sek.
- Throughtputten vil øke som funksjon av belastningen, helt til ett punkt der du ikke vil få mer throughput.
- Finn komponenten som har brukt opp sine ressurser.

Fjerne flaskehals

- Optimalisere applikasjonskoden
- Legge til flere ressurser til komponentene.
- Ta i bruk mellomlager mekanismer (eng. cache)

Mellomlager mekanismer (cache)

- Midlertidig lagre genererte sider og vise disse istedet for å stadig generere samme side om og om igjen.
- Fordeler:
  - forminsker svartiden på en side
  - forminsker ressurser brukt til sidegenerering
- Ulemper:
  - krever *litt* mer av programmerere(n)

Hva kan mellomlagres

- Hele websider
- Deler av en webside (ESI tags i page templates)
- Database forespørsler og forespørsler til andre komponenter.
- Andre ting som bruker mye ressurser

Hvor man kan mellomlagre

- I Webleseren. Både HTML og HTTP header kan forteller når leseren skal fornye den lagrede siden.
- Proxy mellomlager. Alle forspørsler går gjennom en server som kun henter ned en enkelt kopi og gir denne ut til alle som vil ha.
- Content Delivery Network (CDN). Distribuerer kopier av ressurser over ett geografisk område. Forespørler sendes til nærmeste server.
- Server akselerator har mellomlageret nær de genererende komponentene og gir ut kopiene til forespørsler fra andre komponenter i arkitekturen

Hvordan man fornyer mellomlageret

- Hente-metode (Pull)
  - “Utgått dato” på elementer i en side eller hele sider.
  - Henter nytt innhold når det i mellomlageret er utgått.
- Sende-metode (Push)
  - Sende ut ny kopi til en mellomlager server ved jevne mellomrom.

Mellomlagre dynamisk innhold med en server aksellerator

- ESI (Edge Side Includes): Tags i sidemalene som forteller hvilke deler som kan mellomlagres og når delen skal fornyes.
- Implementert i Java/JSP med JESI.