

Komputasi Awan

Oleh:

Tim Penyusun Komputasi Awan
PENS

Materi Ajar

1. Pengenalan terhadap Komputasi Awan
2. Layanan Komputasi Awan
3. Kerangka dan Teknologi Komputasi Awan
4. Integrasi dan Standarisasi Komputasi Awan
5. Keamanan Komputasi Awan
6. Aplikasi Awan pada AWS dan GCP
7. Aplikasi Awan dengan Openstack
8. Big Data dan Analytics pada Komputasi Awan
- 9. Performa, Keandalan dan Ketersediaan Komputasi Awan**
10. Migrasi dan Manajemen Komputasi Awan
11. SKKNI Komputasi Awan
12. Prospek Komputasi Awan

Performa, Keandalan dan Ketersediaan Komputasi Awan

Outline

- Toleransi kegagalan (Fault-tolerant)
- Konsumsi energi
- Evaluasi layanan

Panduhuluan (1)

- Komputasi awan terus berkembang dan semakin kompleks.
- Oleh karenanya penting untuk menjadi stabilitas, ketersediaan dan keandalannya.
- Dengan melakukan *update* dan *upgrade* sistem yang ada menimbulkan adanya kegagalan sistem.

Panduhuluan (2)

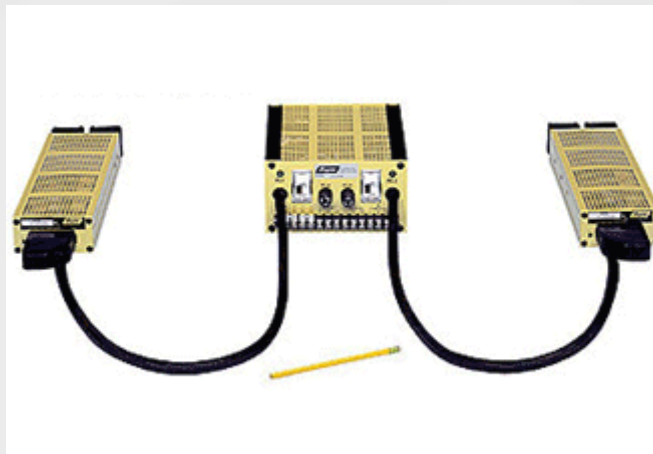
- Misalnya:
 - *Google mengalami kerugian 20% dari pemasukan ketika bereksperimen dimana menyebabkan adanya tambahan delay dari respon sebesar 500ms.
- Agar komputasi awan handal dalam menghadapi kegagalan maka penyedia layanan komputasi awan harus mengimplementasikan sistem *Fault-Tolerance (FT)*.

Jenis-jenis kegagalan

- Diantaranya adalah:
 - (i) Kegagalan *hardware* yang bersifat transien, *intermiten*, maupun permanen.
 - (ii) kesalahan desain dan *software* yang cacat.
 - (iii) kesalahan operator
 - (iv) kesalahan yang disebabkan faktor eksternal.
- Namun yang kegagalan umum yang sering terjadi di *cloud* adalah:
 - *Byzantine*: kegagalan yang terjadi diluar dari kebiasaan dan tidak dapat ditebak (tidak konsisten).
 - *Crash*: kegagalan yang menyebabkan komponen *cloud* berhenti berkerja secara total, misal. *hardisk* rusak atau mati listrik.

Redudansi

- Implementasi dari FT adalah redudansi
- Redudansi → duplikasi dari komponen (*hardware/software*).
- Prinsip kerja: Jika salah satu komponen mengalami kegagalan maka komponen lain menggantikannya.



Validasi FT

- Validasi FT sangat penting untuk memastikan SLA (*service level agreement*).
- Untuk mengujicoba sistem FT maka dapat digunakan *fault-injection*.
- Dan digunakan beberapa aplikasi pengukur performa (latensi, cakupan, tingkat kegagalan) seperti:
 - *Hybrid Automated Reliability Predictor (HARP)* – Duke Univ.
 - *System Availability Estimator (SAVE)* - IBM
 - *Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE)* – Duke Univ.
 - UltraSAN – Illinois Univ.
 - DEPEND – Illinois Univ.

Mengukur FT (1)

- Cara mengukur ketersediaan:

$$availability = \frac{uptime}{uptime + downtime}$$

$$availability(\%) = \frac{agreed\ service\ time - downtime}{agreed\ service\ time} \times 100$$

- Ketersediaan juga dapat diukur dengan melibatkan MTBF, MTTF, MTTR

$$availability = \frac{MTBF}{MTBF + MTTR}$$

MTBF: Mean Time Between Failure

MTTF: Mean Time to Failure

MTTR: Mean Time to Repair

Mengukur FT (2)

- Sedang untuk mengukur kehandalan layanan adalah:

$$\text{service reliability} = \frac{\text{successful responses}}{\text{total requests}} \times 100$$

- Pada beberapa layanan yang dinilai sangat dapat diandalkan maka layanan dinilai sebagai berikut:

$$DPM = \frac{(\text{total requests} - \text{successful transactions})}{\text{total request}} \times 1000000$$

DPM: Defects per million

Strategi FT pada komputasi awan (1)

- Ada dua strategi FT pada komputasi awan:
 - Berbasis checkpoint:

merekam sistem secara periodik pada jangka waktu yang ditentukan, sehingga jika terjadi kegagalan maka posisi *checkpoint* terakhir akan dijadikan acuan untuk pemulihan.
 - FT adaptif:

membantu sistem untuk menjaga dan meningkatkan kehandalan dengan beradaptasi terhadap perubahan lingkungan

Strategi FT pada komputasi awan (2)

Strategy	Faulty tolerance technique	Programming framework	Environment	Faults detected
(Nicolae and Cappello, 2011)	Disk based Checkpoint	MPI	IaaS cloud	Node/network failure
(Hakkarinen and Chen, 2013)	Diskless Checkpoint	NA	HPC	Process/application failure
(Kwak and Yang, 2012)	Checkpoint	Probability analytic framework	Real-time systems	Process failures
(Goiri <i>et al.</i> , 2010)	Checkpoint	Java	Virtual Machine	Node failure
(Malik <i>et al.</i> , 2011)	FTRT (adaptive)	–	Real-time	–
(Sun <i>et al.</i> , 2013)	DAFT (adaptive)	Java	Large scale Cloud	Works on historical failure rate
(Cogo <i>et al.</i> , 2013)	FITCH (adaptive)	Java	Large scale Cloud	–
(Zhang <i>et al.</i> , 2011)	BFTCloud (adaptive)	Java	Voluntary-resource cloud	Byzantine problems
(Zhao <i>et al.</i> , 2010)	LLFT (adaptive)	C++	Middleware	Replication faults
(Ko <i>et al.</i> , 2010)	IFT(adaptive)	Hadoop	Hadoop	Intermediate data faults
(Zheng, 2010)	MFTLL (adaptive)	MapReduce	MapReduce	Replication faults, stragglers detection
(Pannu <i>et al.</i> , 2012)	AAD (adaptive)	–	Local cloud	Discovers future failures

Konsumsi Energi

- Konsumsi energi oleh komputer server tren-nya selalu naik.

Table 25.2 *Estimated average power use of volume, mid-range, and high-end servers (in Watts)*

Type/year	2000	2001	2002	2003	2004	2005	2006
Volume	186	193	200	207	213	219	225
Medium	424	457	491	524	574	625	675
High end	5534	5832	6130	6428	6973	7651	8163

Konsumsi Energi (1)

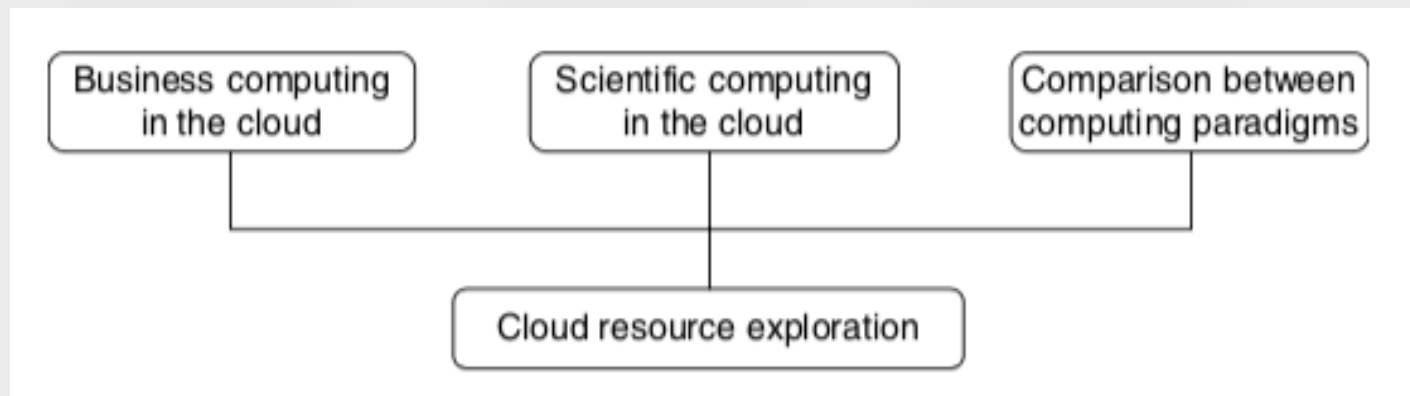
- Optimalisasi konsumsi energi sangat penting untuk masa depan komputasi awan.
- Virtualisasi dapat mengotimalkan kinerja dari server fisik, namun virtualisasi juga memberikan *overhead* yang cukup besar.
- Nilai *overhead* tidak mudah untuk dihitung, karena setiap *hypervisor* memiliki karakter yang berbeda.
- Hal ini yang menyebabkan konsumsi energi yang berbeda-beda.

Konsumsi Energi (2)

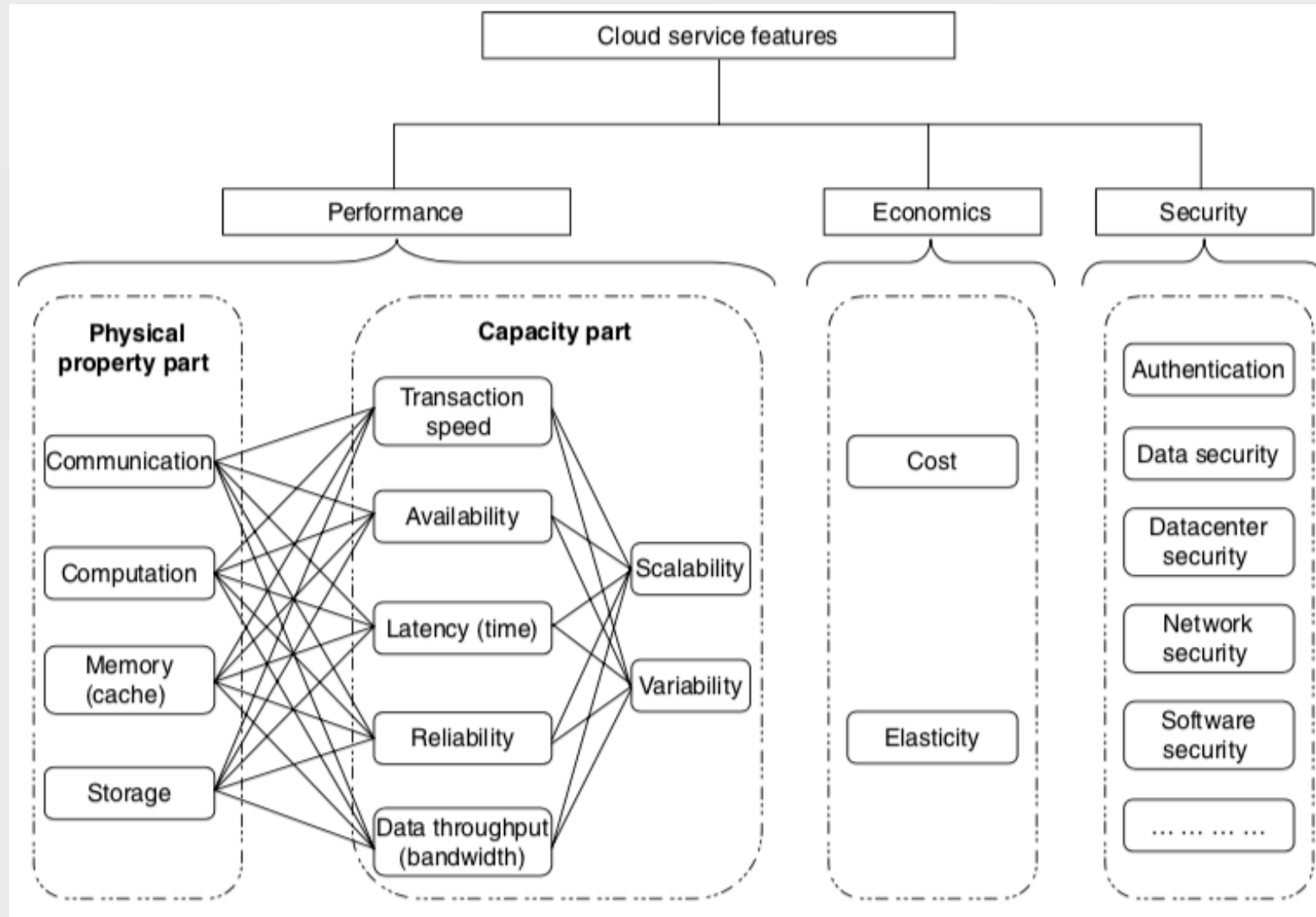
- Contoh: VMware Workstation mengkonsumsi energi 5-6x lebih tinggi dibandingkan VMware native.
- Selain itu fitur virtualisasi seperti migrasi, replikasi data, juga mengkonsumsi energi yang besar.
- Oleh karenanya HW komputer harus selalu ditingkatkan terutama manajemen energi mis. DVFS., penggunaan SSD, dll.

Evaluasi layanan *Cloud*

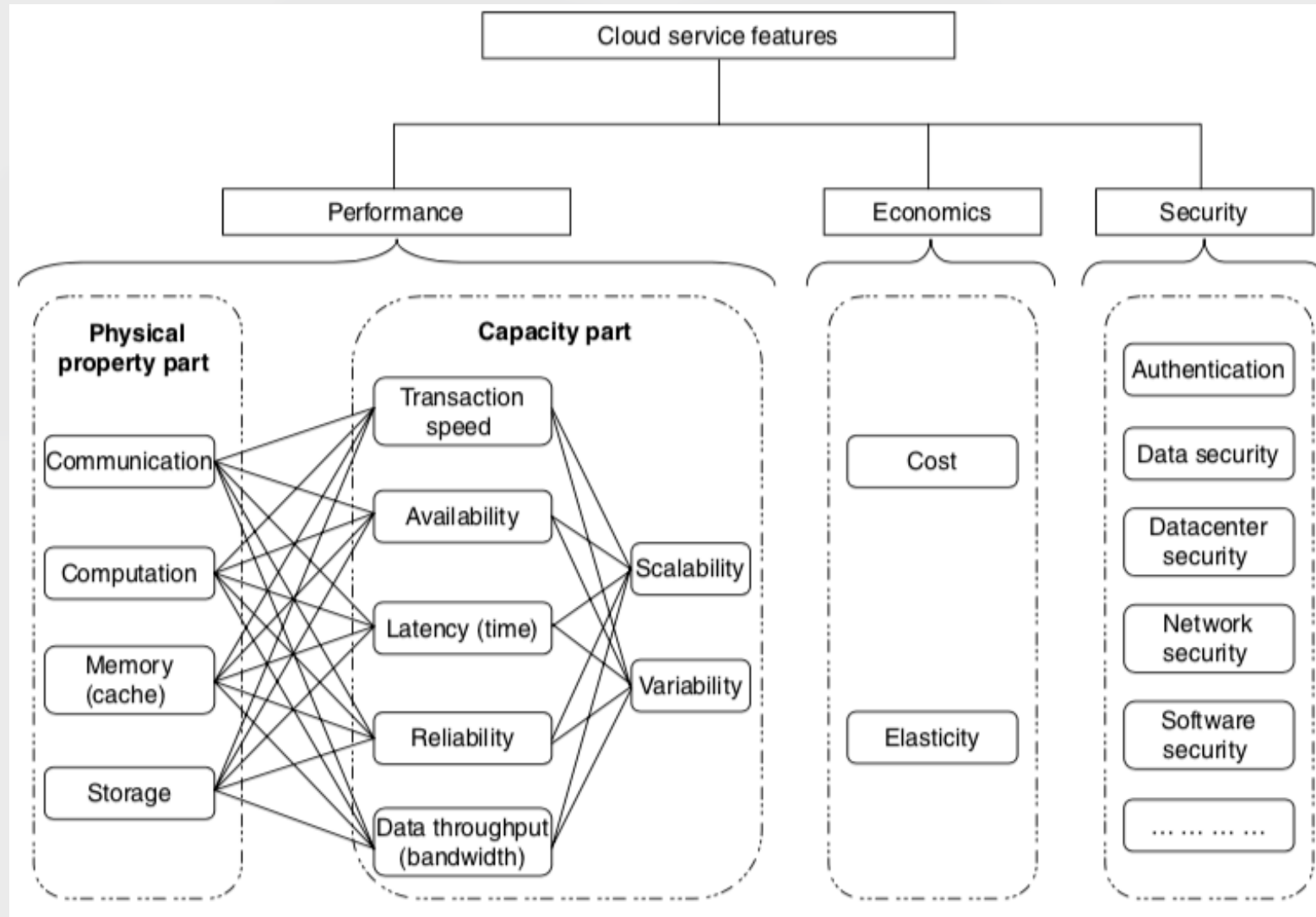
- Evaluasi layanan *Cloud* sangat berguna bagi pengguna dan juga penyedia layanan untuk mengetahui kekurangan dari layanan yang telah disediakan.
- Lebih detail motivasi untuk evaluasi layanan *cloud* adalah (Li *et al*, 2013):



Layanan *Cloud* yang perlu dievaluasi



Layanan *Cloud* yang perlu dievaluasi (1)



Layanan *Cloud* yang perlu dievaluasi (2)

- **Performa**

Biasanya merupakan kombinasi dari evaluasi kondisi fisik dan juga kapasitasnya. Misalnya latensi komunikasi, kehandalan media penyimpanan, dll.

- **Ekonomi**

Aspek komersial dari layanan *cloud* yaitu biaya dan ke-elastisannya. Biaya mengindikasikan seberapa besar kegunaan dari aplikasi. Ke-elastisan mengindikasikan bagaimana sebuah *cloud* dapat berubah/tumbuh dengan mudah.

- **Keamanan**

Misalnya proses autentikasi, validitas pengguna, antivirus, dll.

Evaluasi *Cloud* dengan aplikasi *benchmark*

- Untuk evaluasi *cloud* dapat digunakan beberapa aplikasi seperti berikut ini:
 - Yahoo! Cloud Service Benchmark
 - CloudStone
 - CloudSuite
- Atau:

Table 29.1 Popular traditional benchmarks for evaluating cloud service performance

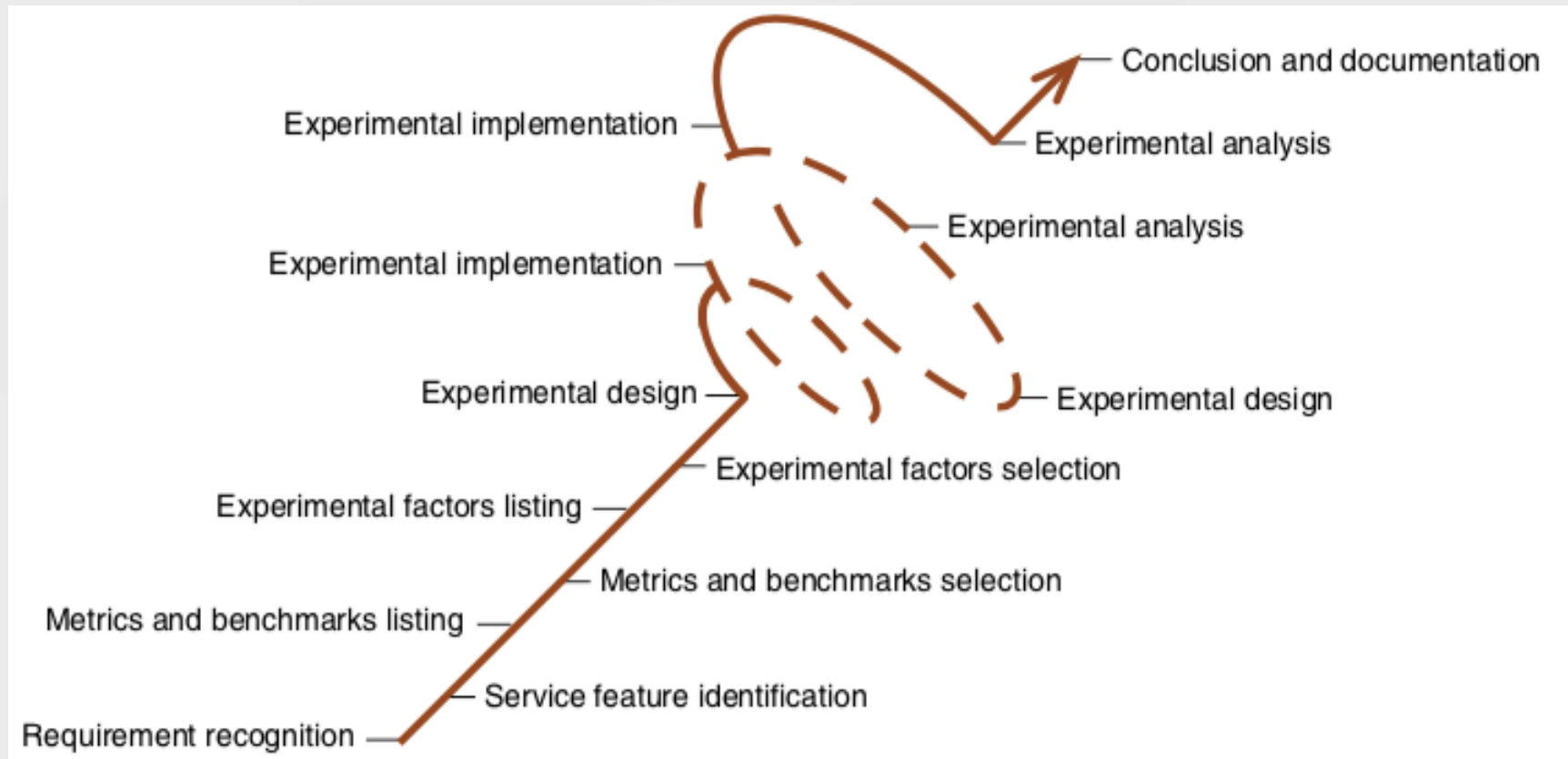
Cloud service performance property	Popular traditional benchmarks
Communication	iperf, ping, Operate/Transfer Data
Computation	HPCC: DGEMM, HPCC: HPL, LMBench
Memory/Cache	HPCC: STREAM
Storage	Bonnie/Bonnie++, IOR, NPB: BT/BT-IO, Operate/Transfer Data
Overall Performance	BLAST, HPCC: HPL, Montage, NPB suite, TPC-W

Metode evaluasi layanan cloud

- Salah satu metodenya adalah ASTAR (Stantchev, 2008)
- Metode lainnya adalah CEEM (*Cloud Evaluation Experiment Methodology*) yang menggabungkan 3 pendekatan evaluasi



Prosedur evaluasi dengan CEEM (1)



Referensi

- Bilal, Kashif, et al. "Fault Tolerance in the Cloud." Encyclopedia of Cloud Computing (2016): 291-300.
- Marinescu, Dan C. "Cloud energy consumption." Encyclopedia of Cloud Computing (2016): 301.
- Li, Zheng, Liam O'Brien, and Rajiv Ranjan. "Cloud Service Evaluation." Encyclopedia of Cloud Computing (2016): 349.

TERIMA KASIH