SURVEI TEKNIK MONITORING BERBASIS INSTRUMENTASI PADA LEVEL HOST, PLATFORM, DAN SERVICE PADA ARSITEKTUR MICROSERVICE

Achmadaniar Anindya Rhosady[1)], Fuad Dary Rosyadi[2)], dan Lucas Susanto[3)]
[1, 2,3,)]Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, Sukolilo, Surabaya, Indonesia 60111
e-mail: achmadaniar.19051@mhs.its.ac.id[1)], rosyadi.19051@mhs.its.ac.id[2)], lucas.19051@mhs.its.ac.id[3)]

ABSTRAK

Microservice adalah arsitektur aplikasi yang bertujuan untuk membagi satu aplikasi yang besar menjadi beberapa aplikasi yang lebih kecil. Arsitektur ini menyederhanakan proses development, deployment, dan manajemen aplikasi. Namun, arsitektur ini memiliki kekurangan pada aspek kompleksitas dan observabilitasnya sehingga proses monitoring microservice menjadi cukup sulit. Agar proses monitoring microservice menjadi lebih praktis, diperlukan klasifikasi instrumentasi-instrumentasi monitoring yang jelas. Kami melakukan survei terhadap metode klasifikasi teknik monitoring pada arsitektur microservice. Metode monitoring ini dibagi menjadi tiga bagian yaitu host level, platform level, dan service level. Dalam penelitian ini, kami menguraikan apa saja instrumen terkini yang sedang digunakan dalam proses monitoring pada setiap level. Hubungan antara aspek tujuan, kebutuhan, dan stakeholder juga dipaparkan.

***Kata Kunci**: Metode monitoring, instrumentasi, microservice.*

INSTRUMENTATION-BASED MONITORING TECHNIQUES SURVEY ON HOST, PLATFORM, AND SERVICE LEVEL IN MICROSERVICE ARCHITECTURE

Achmadaniar Anindya Rhosady[1)], Fuad Dary Rosyadi[2)], dan Lucas Susanto[3)]
[1, 2,3,)]Department of Informatics, Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, Sukolilo, Surabaya, Indonesia 60111
e-mail: achmadaniar.19051@mhs.its.ac.id[1)], rosyadi.19051@mhs.its.ac.id[2)], lucas.19051@mhs.its.ac.id[3)]

ABSTRACT

Microservice is an application architecture that separates one big application into smaller ones. The architecture simplifies development, deployment, and management process. However, the architecture is quite complex thus the monitoring process becomes much more challenging. Classifications for the instrumentations that are used in the monitoring process is needed to achieve better practicality for the administrators. We surveyed the monitoring technique classification method in microservice architecture. The method is divided into three levels. They are host level, platform level, and service level. In this paper, we present the latest instruments that are being used in the monitoring process in each level. Correlation between the goals, needs, and stakeholder is also presented.

***Keywords**: Instrumentation, microservice, monitoring method.*

I. PENDAHULUAN

MICROSERVICE adalah arsitektur aplikasi yang bertujuan untuk membagi satu aplikasi yang besar menjadi beberapa aplikasi yang lebih kecil. Arsitektur ini menyederhanakan proses *development*, *deployment*, dan manajemen aplikasi. Kelebihan arsitektur ini adalah arsitektur ini bersifat *loose-coupled* karena adanya pembagian fungsi yang jelas (*separation of concern*), handal (*reliable*) jika salah satu aplikasi mengalami *crash* maka aplikasi tersebut tidak akan mempengaruhi jalannya aplikasi-aplikasi yang lain secara signifikan, dan aplikasinya berukuran jauh lebih kecil.

Pada penerapannya dalam sistem berskala besar, arsitektur ini memiliki kekurangan pada aspek kompleksitas dan observabilitas. *Microservice* memiliki banyak aplikasi sehingga sulit untuk memonitoring kondisinya secara *real time*. Di sisi lain, kegagalan dalam satu *microservice* dapat berdampak pada *microservice* lain. Oleh karena itu, sangat sulit untuk menentukan serangkaian *microservice* yang dapat menghambat kinerja dalam melayani permintaan pengguna. Memonitor *microservice* dapat menjadi faktor kunci untuk mendeteksi kegagalan layanan lebih awal karena kegagalan pada *microservice* dapat terjadi kapan saja. Oleh karena itu, diperlukan suatu

Teknik monitoring *microservice* untuk mempermudah pelacakan titik-titik kegagalan dan kondisi-kondisi anomali agar aplikasi-aplikasi *microservice* dapat terus berjalan dengan baik.

Untuk dapat menentukan teknik *monitoring* yang akan digunakan dalam sistem, pengetahuan mengenai arsitektur *microservice* menjadi sangat penting. Pengetahuan berasal dari informasi sedangkan informasi berasal dari data. Data didapat dari hasil keluaran sistem *monitoring*. Sistem ini menggunakan instrumen-instrumen untuk mengidentifikasi fitur-fitur status sistem yang beragam. Pengetahuan yang telah diketahui menentukan apa, bagaimana dan di mana teknik monitoring tersebut akan digunakan dalam sistem. Selain itu, pengetahuan juga sangat menentukan *stakeholder* yang akan terlibat dalam proses *monitoring*.

Pada penerapan arsitektur *microservices* dapat dibagi menjadi 3 level, yaitu level *host*, *platform*, dan *service*. Pembagian ini didasarkan pada level *monitoring* data yang ada pada *microservices*. *Monitoring* pada level *host* akan memberikan gambaran sistem terkait data infrastruktur seperti jumlah *host* yang tersedia, konsumsi CPU dan memori, serta pemetaan *service* pada infrastruktur. Monitoring pada level *platform* akan memberikan gambaran sistem terkait data *runtime* yang spesifik dari suatu *service* yang sedang berjalan seperti *response time, throughput*, dan *failure rate* dari suatu *service*. Sedangkan monitoring pada level *service* akan memberikan gambaran tentang data terkait interaksi antar *service* yang sedang beroperasi seperti alur komunikasi dari suatu *service*.

Penentuan level instrumentasi untuk proses monitoring akan sangat berpengaruh terhadap data seperti yang akan diperoleh apa dan bagaimana prosesnya. Sehingga pada paper ini akan dibahas mengenai klasifikasi proses monitoring berdasarkan instrumentasi arsitektur *microservice* yang dibagi menjadi tiga bagian yaitu host *level*, platform *level*, dan service *level*. Kemudian akan dijelaskan mengenai kelebihan dan kekurangan dari masing masing literatur serta mendapatkan korelasi dengan *stakeholder* yang terkait. Selain itu juga akan dilakukan pemetaan berdasarkan data monitoring yang diperoleh dari masing masing metode.

Sistematika penulisan paper ini adalah sebagai berikut. Pertama, penelitian terkini terkait metode *monitoring* pada arsitektur *microservice* diuraikan. Berikutnya, teknik *monitoring* berbasis instrumentasi dijelaskan secara detail. Pembagian level klasifikasi juga akan dipaparkan pada bagian ini. Setelah itu, kami melakukan survei terhadap instrumen-instrumen yang digunakan dalam proses *monitoring* pada penelitian terkini. Terakhir, kesimpulan mengenai teknik *monitoring* ini akan diuraikan. Diskusi juga akan dimuat pada bagian ini.

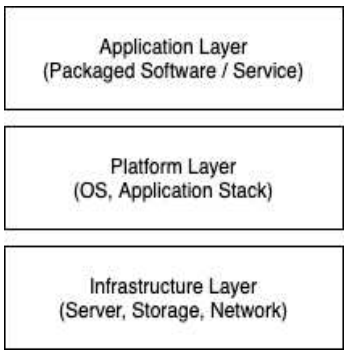
II. PENELITIAN TERKAIT

Terdapat beberapa penelitian yang telah dilakukan terkait dengan *monitoring* dalam arsitektur *microservice*. Sowmya dkk. (2014) melakukan sebuah survey mengenai arsitektur dari sebuah cloud. Pada survey tersebut dijelaskan bahwa dalam arsitektur cloud terdiri dari 3 layer, yaitu infrastruktur, platform dan aplikasi. Layer infrastruktur sebagai lapisan dasar, platform sebagai antarmuka antara aplikasi dan perangkat keras dan akhirnya software lapisan atas menjadi ujung depan yang dengannya pengguna berinteraksi [1]. Namun pada arsitektur *microservice*, *application layer* pada arsitektur tersebut juga bisa disebut sebagai *service layer*. Arsitektur layer pada paper ini dapat diilustrasikan seperti pada Gambar 1.

Haselbock dkk. (2017) melakukan sebuah penelitian untuk membuat sebuah pedoman untuk mengambil keputusan dalam mendesain sistem monitoring pada *microservice*. Dalam penelitian tersebut, Haselbock merumuskan model untuk mendapatkan data dari proses *monitoring*, pemrosesan data untuk menjadi informasi, hingga pada tahapan presentasi informasi pada stakeholder yang terkait. Haselbock membagi fokus penelitian pada 3 aspek, yaitu Tujuan, Kebutuhan, dan *Stakeholder* [2]. Namun, pada penelitian ini tidak ditampilkan secara detail kelebihan dan kekurangan literatur yang digunakan sebagai referensi untuk mendapatkan model tersebut.

Selanjutnya, Haselbock dkk. (2018) melakukan sebuah survei dengan cara melakukan interview terhadap *stakeholder* yang terlibat dalam sistem *microservice*. Survei yang dilakukan untuk mengetahui desin area dari sebuah *microservice*, tingkat kepentingannya, serta tantangannya. *Stakeholder* yang terlibat dalam survey ini ada terbagi menjadi 6 bidang, yaitu *Developer, Architect, Operation, Quality Assurance, Researcher,* dan *Consultant*. Dalam penelitian ini disebutkan bahwa design area dari *monitoring* dan *logging* pada *microservice* memiliki tingkat kepentingan paling tinggi [3]. Namun pada penelitian ini, survei hanya terbatas pada hasil interview dari *stakeholder* saja.

Soldani & Tamburri (2018) melakukan *literature review* mengenai kesulitan dan manfaat dari implementasi sistem *microservices*. Pada paper ini, taksonomi dilakukan dengan membagi menjadi 3 fase utama, yaitu *Design, Development,* dan *Operation*. Pada taksonomi berdasarkan kesulitan, fase spesifik terdiri dari desain arsitektur dan sekuritas, *Development microservice*, *testing* dan *storage*, serta *Operation management*, *monitoring* dan konsumsi sumberdaya. Serta pada taksonomi manfaat, fase spesifik terdiri dari desain arsitektur, pola, dan sekuritas,



Gambar 1. Arsitektur layer *microservice cloud.*

Development microservice, *storage*, dan *testing*, serta *Operation deployment* dan *management* [4]. Namun pada penelitian ini tidak menyebutkan relasi dengan *stakeholder* yang terkait dalam referensi literatur yang dibahas.

III. METODE MONITORING BERBASIS KLASIFIKASI

Klasifikasi monitoring berbasis instrumentasi adalah klasifikasi yang menggunakan parameter-parameter fitur-fitur status sistem sebagai dasar untuk membuat pengetahuan kondisi aplikasi. Tidak seperti penelitian yang sebelumnya. klasifikasi ini dapat memperlihatkan hubungan antara kondisi aplikasi dengan jenis serta klasifikasi level yang dipilih. Dengan diketahuinya hal tersebut, monitoring *microservice* dapat dilakukan dengan lebih tepat sasaran sesuai dengan kebutuhan serta *stakeholder* yang terlibat.

Pada metode *monitoring* ini, ada tiga buah jenis klasifikasi, yaitu *host level*, *platform level*, *dan service level*. Level ini berkaitan dengan peletakan agen *monitoring* dalam sebuah infrastruktur *microservice* untuk mendapatkan data seperti apa yang akan diperoleh dan diproses untuk dijadikan sebuah informasi. Visualisasi level ini dapat dilihat pada Gambar 2. Penjelasan lebih lanjut mengenai level-level ini akan dijelaskan pada subbab berikutnya.

A. Host Level Instrumentation

Host level membutuhkan sebuah agen *monitoring* yang diletakkan pada setiap host dari infrastruktur microservice. Peletakan pada host level bertujuan untuk memperoleh data yang berkaitan dengan *runtime* host. Dalam hal ini data yang berkaitan dengan *runtime host* dapat berupa *response time*, *failure rate*, *throughput*, beban CPU, dan penggunaan memori. Peletakan pada level *host* juga memungkinkan untuk dapat mendeteksi secara otomatis apabila terdapat sebuah *microservice* baru yang ditambahkan pada sebuah host. Kelebihan yang dimiliki pada arsitektur ini adalah dari perspektif *agent deployment*, karena hanya perlu meletakkan pada *host* sehingga pengembang *service* tidak perlu terlibat dalam dengan instrumentasinya. Namun kekurangan yang dimiliki pada peletakan agen *monitoring* pada level *host* adalah keterbatasan untuk memperoleh informasi yang spesifik dari suatu *service*.

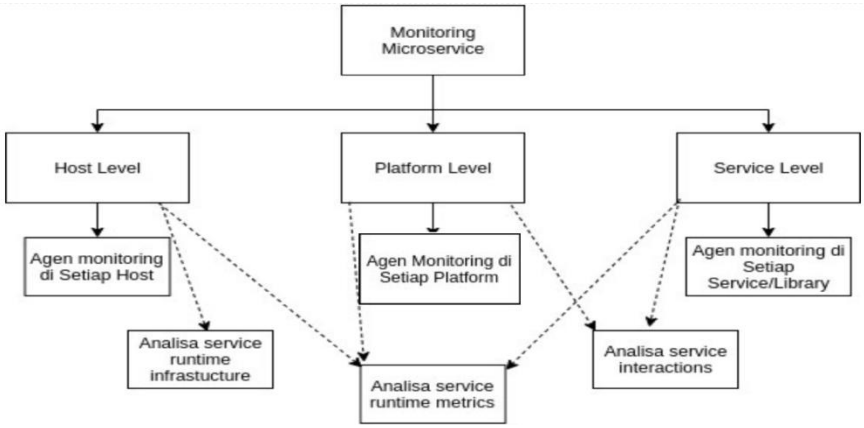
B. Platform Level Instrumentation

Platform level membutuhkan sebuah agen *monitoring* yang diletakkan pada setiap *platform* dari suatu *host microservice*. Agen *monitoring* pada level *platform* dapat lebih spesifik pada teknologi yang digunakan pada *platform microservice* jika dibandingkan dengan *host level*, sehingga agen tidak terkait secara langsung dengan sistem operasi yang digunakan. Sebagai contoh agen untuk *Java Virtual Machine* (JVM) misalnya, dapat mengumpulkan data tentang pengumpulan *log* dan penggunaan memori dalam JVM. Namun, kebutuhan satu agen untuk setiap *platform* membutuhkan penyebaran dan pengoperasian beberapa agen pada setiap *host*.

C. Service Level Instrumentation

Service level membutuhkan sebuah agen monitoring yang diletakkan pada setiap *service*. Hal ini memungkinkan untuk mendapatkan informasi yang spesifik mengenai *service* yang sedang berjalan, seperti matriks *runtime service* dan interaksi antar *service*. Agen yang berada pada *service level* lebih spesifik jika dibandingkan dengan *host platform level* dan dapat berupa *library* yang di-*deploy* pada setiap *service*. Sehingga dalam melakukan *monitoring* pada *level service* akan menambah beban pada *service developer*.

Dari klasifikasi di atas kemudian dilakukan pencarian literatur yang sesuai dengan topik yang diangkat pada paper ini. Adapun kata kunci yang digunakan untuk mencari literatur tersebut adalah "*microservice monitoring*", "*microservice analysis*", "*microservice architecture*", dan "*microservice design*". Distribusi literatur yang telah dipilih dan akan dibahas pada paper ini dapat dilihat pada Tabel I.



Gambar 2. Pembagian klasifikasi *monitoring*.

TABEL I
TABEL DISTRIBUSI KELAS *MONITORING MICROSERVICE* BERBASIS INSTRUMENTASI.

Kelas	Penulis
Host level	Yang & Huang (2019)
	Thalheim dkk.. (2017)
	Edoardo Fadda, Pierluigi Plebani (2016)
	Canali & Lancellotti, (2014)
Platform level	Noor dkk. (2019)
	Kargar & Hanifizade (2018)
	Pina, Correia, Filipe, Araujo, & Cardroom (2018)
Service level	Cinque, Della Corte, & Pecchia, (2019)
	Gkikopoulos (2019)
	Toffeti dkk. (2015

IV. HASIL SURVEI

Survei dilakukan pada penelitian-penelitian terkini mengenai proses *monitoring microservice*. Survei dilakukan dengan batasan topik monitoring berdasarkan instrumentasi dari arsitektur microservice. Ada tiga buah klasifikasi instrumentasi yaitu *host level*, *platform level*, dan *service level*.

A. Host Level Instrumentation

Yang & Huang (2019) mengusulkan *tools* untuk *monitoring* pada *microservice* berbasis *Openstack* yang dinamai *Openstack Reporter*. Pada teknik ini, agen diletakkan pada sebuah *host* dengan Kubernetes. Teknik ini bertujuan untuk mendapatkan data *monitoring* yang nyaman dan mudah digunakan oleh administrator. *Openstack Reporter* terbagi menjadi 3 aplikasi, yaitu *openstack-exporter*, *prometheus*, dan *grafana* [5]. Penelitian ini memiliki kelebihan dalam hal kemudahan dalam proses *deployment*. Kekurangan dari *tools* ini adalah tidak menyediakan informasi spesifik terkait aplikasi. *Stakeholder* yang terlibat dalam paper ini adalah *system administrator*.

Thalheim dkk. (2017) mengusulkan teknik monitoring pada *microservice* yang dinamai SIEVE. Teknik ini diperuntukkan untuk sistem terdistribusi berbasis *microservice* yang berbasis *Openstack* dan *ShareLatex* . Agen ini berbentuk *platform* yang di-*deploy* pada sebuah *host*. Teknik ini memungkinkan untuk mendapatkan informasi dari hasil *monitoring* dari sistem secara otomatis. Hasil *monitoring* yang diperoleh dari teknik ini adalah metrik *runtime* pada *host*, seperti *CPU*, *memory*, *bandwidth*, dan *Disk I/O* [6]. Kelebihan dari teknik ini adalah dapat mengurangi jumlah metrik hingga 100 kali untuk membuat statistik sehingga dapat mengurangi *overhead* yang digunakan untuk proses *monitoring*. Selain itu teknik yang diusulkan juga bersifat *unsupervised.* Namun untuk kekurangannya, *traffic traces* yang dihasilkan dari teknik ini tidak dapat menentukan dimana *stress point* dari aplikasi karena tidak bersifat spesifik. Sedangkan untuk *stakeholder* yang terlibat dalam implementasi teknik ini adalah application *developer* dan *operations*.

Fadda et. al (2016) mengusulkan teknik optimasi dalam *monitoring* pada *distributed system* berbasis multi objektif. Teknik yang diusulkan menggunakan agen yang di-*deploy* pada sebuah *host*. Agen ini berfungsi sebagai *broker* yang dapat menentukan informasi fitur seperti apa yang ingin diperoleh dan mengarahkannya pada VM aplikasi yang spesifik [7]. Kelebihan dari teknik *monitoring* ini adalah dapat mengurangi biaya untuk mendapatkan informasi, serta tetap menjaga kualitas dari hasil *monitoring*. kekurangan dari teknik ini adalah

sistem *monitoring* memerlukan peran dari *stakeholder operations* yang cukup intens. selain dari *operations stakeholder* yang terlibat pada teknik *monitoring* ini adalah *developer*.

Canali & lancellotti (2014) mengusulkan sebuah teknik *monitoring* yang adaptif terhadap skalabilitas sistem. Teknik ini melakukan *monitoring* dan manajemen berbasis *cluster*. *Clustering* dilakukan dengan melihat karakteristik dari VM yang melakukan klasifikasi dengan menggunakan *Fuzzy Logic*. Proses *clustering* dilakukan untuk menjaga kemampuan skalabilitas dari sistem sehingga dapat memudahkan proses menambahkan atau mengurangi VM yang dinamis [8]. Namun kekurangan dari teknik ini adalah proses *clustering* membutuhkan waktu delay yang cukup lama untuk dapat melakukan klasifikasi yang akurat. *Stakeholder* yang terlibat pada teknik ini adalah *operations*.

B. Platform Level Instrumentation

Noor dkk. (2019) mengusulkan sebuah *framework* untuk *monitoring* pada *microservice* yang heterogen. Teknik yang diusulkan yaitu M3 *(Multi-microservices, Multi-virtualization, Multi-cloud).* Teknik ini memungkinkan untuk melakukan proses monitoring dari lingkungan yang heterogen yang kompleks. Agen *monitoring* diletakkan pada *cloud platform* yang ada pada setiap VM. Agen ini berfungsi untuk mendapatkan data dari *microservice* yang sedang berjalan [9]. Kekurangan dari teknik *monitoring* ini adalah perlunya *resource* yang cukup signifikan dari agen yang digunakan. Sementara *stakeholder* yang terlibat dalam teknik ini adalah *developer* dan *operations*.

Kargar & Hanifizade (2018) mengusulkan sebuah teknik *regression test automation* pada arsitektur *microservice*. Teknik ini memanfaatkan *Continuous Delivery* untuk membantu *development process* dari *microservices*. Sedangkan *regression test* bertujuan untuk memastikan *reliability* dari sistem *microservice* [10]. Kelebihan dari teknik yang diusulkan yaitu dapat mengurangi *production cost* serta memudahkan proses *deployment*. Sementara kekurangannya yaitu sistem masih terbatas pada layanan yang deterministik. *Stakeholder* yang terlibat dalam teknik ini adalah *developer* dan *operations*.

Pina dkk. (2018) mengusulkan sebuah teknik *non-intrusive monitoring system* pada *microservice*. Teknik yang diusulkan menggunakan sistem *log gateway* antar *microservice* untuk melakukan *monitoring* [11]. Teknik ini memungkinkan untuk mendapatkan metrik performa tanpa memodifikasi *source code*. Kekurangan dari dari teknik ini yaitu masih belum dapat menentukan deteksi anomali secara otomatis. *Stakeholder* yang terlibat pada teknik ini adalah *developer*, *operations*, dan *system administrator*.

C. Service Level Instrumentation

Cinque, Della Corte, & Pecchia (2019) mengusulkan sebuah teknik monitoring pada *microservice* berbasis *passive tracing* dan *log analysis. Passive tracing* digunakan untuk memperoleh interaksi antar *microservice*, sedangkan *log analysis* digunakan untuk proses ekstraksi log microservice untuk mendapatkan informasi. teknik ini bersifat *non-intrusive* sehingga tidak perlu melibatkan perubahan pada aplikasi [12]. Namun kekurangan dari sistem ini adalah sangat terbatas pada data yang ada, sehingga perlu banyak melakukan percobaan terhadap macam-macam studi kasus. *Stakeholder* yang terlibat dalam teknik ini adalah *system administrator.*

Gkikopopoulos (2019) mengusulkan sebuah teknik distribusi dan eksploitasi data pada sistem *microservice*. Teknik ini melakukan proses *mining* dan eksploitasi terhadap data yang ada. Proses ini digunakan untuk mendapatkan informasi yang lebih banyak untuk *developer* dan *manager*. Pada penelitian ini, menggunakan studi kasus *microservice artefact observatory*. Data yang dieksploitasi didapatkan pada proses *monitoring* yang dilakukan pada *artefact marketplaces* [13]. Kelebihan yang dimiliki teknik ini adalah dapat mendapatkan informasi yang lebih akurat dan terarah. namun kekurangan dari teknik ini juga masih sangat bergantung pada data.

Toffetti dkk. (2018) mengusulkan sebuah arsitektur *microservice* pada *cloud* yang bersifat *scalable* dan memiliki kemampuan untuk *self-management*. Data *monitoring* pada teknik ini digunakan sebagai proses *auto-scaling decision*. Data *monitoring* yang didapatkan berupa *runtime metric* pada *service* [14]. Kelebihan yang dimiliki oleh teknik ini adalah mampu melakukan *self-management* dengan *decision making* berbasis data. namun kekurangan dari teknik ini belum mampu diimplementasikan pada lingkungan yang heterogen. *Stakeholder* yang terlibat pada metode ini yaitu system *administrator* dan *developer.*

V. DISKUSI DAN PEMBAHASAN

Dari hasil review pada bab IV selanjutnya akan dirangkum sedemikian rupa sehingga dapat disajikan dalam bentuk tabel seperti pada Tabel II. Pada tabel tersebut menunjukkan penulis serta metode yang diusulkan. Dari masing masing metode tersebut dipaparkan secara ringkas kelebihan dan kekurangannya serta data yang diperoleh dari teknik yang digunakan. Adapun data yang diperoleh dari proses monitoring dibagi menjadi 3 jenis, yaitu *Runtime Infrastructure Metrics, Service Runtime Metrics* dan *Service Interactions.* Penentuan level instrumentasi sangat menentukan data yang akan diperoleh dari teknik monitoring yang digunakan.

Instrumentasi pada *host level* lebih mengutamakan untuk mendapatkan data berupa penggunaan *Runtime Infrastructure Metrics* seperti jumlah *host* yang tersedia, konsumsi CPU dan memori pada suatu host, serta pemetaan suatu service dalam suatu infrastruktur. Sedangkan instrumentasi pada *platform level data* yang diperoleh lebih spesifik pada *Service Runtime Metrics*, seperti *response time, failure rate,* dan *throughput*. Namun pada *level platform* tidak menutup kemungkinan untuk mendapatkan informasi runtime pada *host level*. Sementara instrumentasi pada service level data yang diperoleh lebih berkaitan dengan *Service Interaction* seperti alur komunikasi antar service. Selain itu juga pada *level service* dapat diperoleh data yang mendukung untuk dilakukan *Root Cause Analysis.*

TABEL II
TABEL DISTRIBUSI KELAS MONITORING MICROSERVICE PADA LEVEL HOST.

Penelitian	Metode	Kelebihan	Kekurangan	Monitoring Data
Yang & Huang (2019)	Openstacks Reporter: *tools* untuk melakukan *monitoring* microservice berbasis *openstack*	menyediakan *tools* yang mudah dan nyaman untuk administrator sistem	tidak menyediakan informasi spesifik terkait aplikasi	*Runtime Infrastructure Metrics*
Thalheim dkk. (2017)	SIEVE: teknik *monitoring microservices* dengan agen berbasis platform dengan sistem reduksi matriks dan mendukung *auto scaling* dan *Root Cause Analysis*	peningkatan performa dapat bersifat *unsupervised*	tidak dapat menentukan stress point pada aplikasi memerlukan beban tambahan saat proses *deployment*	*Runtime Infrastructure Metrics*
Edoardo Fadda, Pierluigi Plebani (2016)	Teknik optimasi *monitoring* pada sistem terdistribusi berbasis multi objektif	mengurangi biaya *monitoring* menjaga kualitas hasil *monitoring*	memerlukan peran *stakeholder operations* yang intens	*Runtime Infrastructure Metrics*
Thalheim dkk. (2017)	SIEVE: teknik *monitoring microservices* dengan agen berbasis platform dengan sistem reduksi matriks dan mendukung *auto scaling* dan *Root Cause Analysis*	peningkatan performa dapat bersifat *unsupervised*	tidak dapat menentukan stress point pada aplikasi memerlukan beban tambahan saat proses *deployment*	*Runtime Infrastructure Metrics*
Canali & Lancellotti (2014)	Teknik *monitoring* yang adaptif terhadap skalabilitas sistem dengan menggunakan *clustering* dan *classification*	menjaga skalabilitas sistem	membutuhkan *delay* yang cukup lama untuk mendapatkan hasil yang akurat	*Runtime Infrastructure Metrics*
Noor et al (2019)	M3: *framework monitoring microservice* pada aplikasi berbasis *cloud* yang heterogen	memungkinkan *monitoring* pada lingkungan heterogen	memerlukan resource tambahan pada setiap agen yang diletakkan pada *platform*	*Runtime Infrastructure Metrics* *Service Runtime Metrics*
Kargar & Hanifizade (2018)	Teknik *regression test automation* pada arsitektur *microservice*	mengurangi *production cost* memudahkan proses *deployment*	terbatas pada *deterministic services*	*Service Runtime Metrics*
Pina, Correia, Filipe, Araujo, & Cardroom (2018)	Teknik *non-intrusive monitoring system* pada arsitektur *microservice* berbasis *activity log*	mengurangi *production cost* bersifat *non-intrusive*	deteksi anomali belum otomatis	*Resource Runtime Infrastructure Metrics* *Service Runtime Metrics*
Cinque, Della Corte, & Pecchia, (2019)	Teknik monitoring pada microservice berbasis *passive tracing* dan *log analysis*	bersifat *non-intrusive* dapat memonitor hingga level interaksi antar *service*	sangat bergantung pada data	*Service Interactions*
Gkikopoulos (2019)	Teknik distribusi dan eksploitasi data pada sistem *microservice*	dapat menghasilkan informasi yang lebih akurat	sangat bergantung pada data	*Service Interactions*
Toffeti et al (2015)	Teknik arsitektur *microservice* pada cloud yang bersifat *scalable* dan memiliki kemampuan untuk *self-management*	*self-management* otomatis berbasis data	sangat bergantung pada data tidak dapat diimplementasikan dalam lingkungan heterogen	*Service Interactions*
Cinque, Della Corte, & Pecchia, (2019)	Teknik monitoring pada microservice berbasis *passive tracing* dan *log analysis*	bersifat *non intrusive* dapat memonitor hingga level interaksi antar *service*	sangat bergantung pada data	*Service Interactions*

Monitoring pada level *host* bermanfaat untuk sistem peringatan ketika suatu *resource* yang digunakan telah melampaui batas dari penggunaan normal serta menjaga aspek high *availability* dari sistem *microservice*. *Monitoring* pada level *platform* bermanfaat untuk membuat *system reporting* dari sistem *microservice* sebagai pertimbangan untuk menjaga aspek skalabilitas dari sistem microservice. Sedangkan *monitoring* pada level *service* bertujuan untuk proses *tracing* yang berguna untuk proses menentukan dimana letak akar permasalahan dari kegagalan dari sebuah sistem *microservice* sehingga dapat mempercepat penyelesaian masalah.

Selain data yang diperoleh, level instrumentasi juga menentukan *stakeholder* yang terlibat dalam sistem *microservice*. Pada level instrumentasi *host*, *stakeholder* yang memiliki keterkaitan diantaranya adalah *developer* dan *operations*. Pada level instrumentasi *platform*, *stakeholder* yang memiliki keterkaitan diantaranya adalah *service developer* dan *system administrator*. Sedangkan pada level instrumentasi *service*, *stakeholder* yang terlibat diantaranya adalah *service developer*, *system administrator*, dan *manager*.

Pada *monitoring* level *host* dapat dikembangkan penelitian lebih lanjut tentang *framework* monitoring *microservice* yang tidak membutuhkan *resource* tambahan agar tidak mengurangi *resource* yang tersedia untuk *microservice*. *Monitoring* pada level *platform* juga dapat dikembangkan lebih lanjut tentang *framework* yang dapat melakukan *monitoring microservice* yang lebih bersifat umum dan tidak terbatas pada *service* yang dijalankan.

Sedangkan pada *monitoring* pada level *service* diperlukan adanya *framework* yang dapat melakukan monitoring interaksi antar *microservice* tanpa memerlukan data pada *microservice* tersebut. Selain itu penerapan *machine learning* pada *monitoring microservice* juga merupakan topik penelitian yang masih terbuka lebar.

VI. KESIMPULAN

Proses *monitoring* pada *microservice* mendapat paling banyak perhatian dalam *design area* dari *microservices*. Pada paper ini, kami melakukan survei terhadap literatur mengenai metode *monitoring* pada *microservice*. Dari paper tersebut kami melakukan klasifikasi menjadi 3 kelas berdasarkan instrumentasi yang digunakan dari masing masing paper, yaitu *host level*, *platform level*, dan *service level*. Pada setiap kelas, dipaparkan mengenai deskripsi dari metode *monitoring* yang digunakan serta menganalisis kelebihan, kekurangan dan data yang diperoleh dari proses monitoring kemudian menampilkannya ke dalam tabel perbandingan. Klasifikasi berdasarkan instrumentasi sangat menentukan data yang akan diperoleh dan *stakeholder* yang akan terlibat di dalamnya.

Monitoring pada *level host* akan memberikan gambaran sistem terkait data *runtime metrics* dari infrastruktur serta cenderung melibatkan *stakeholder* bagian *Operations*. *Monitoring* pada level *platform* akan memberikan gambaran sistem terkait data *runtime* yang spesifik dari suatu *service* yang sedang berjalan serta cenderung melibatkan *stakeholder* bagian *System Administrator*. Sedangkan *monitoring* pada level *service* akan memberikan gambaran tentang data terkait interaksi antar *service* melalui yang sedang beroperasi seperti alur komunikasi antar *service* dan cenderung melibatkan *stakeholder* bagian *Developer* dan *System Administrator*. Dengan dilakukannya survei ini diharapkan mampu membantu pembaca untuk menentukan teknik *monitoring* yang cocok untuk digunakan dalam *microservices* sesuai dengan kebutuhan.

Pada *monitoring* level *host* dapat dikembangkan penelitian lebih lanjut tentang *framework* monitoring *microservice* yang tidak membutuhkan *resource* tambahan agar tidak mengurangi *resource* yang tersedia untuk *microservice*. *Monitoring* pada level *platform* juga dapat dikembangkan lebih lanjut tentang *framework* yang dapat melakukan *monitoring microservice* yang lebih bersifat umum dan tidak terbatas pada *service* yang dijalankan. Sedangkan pada *monitoring* pada level *service* diperlukan adanya *framework* yang dapat melakukan monitoring interaksi antar *microservice* tanpa memerlukan data pada *microservice* tersebut. Selain itu penerapan *machine learning* pada *monitoring microservice* juga merupakan topik penelitian yang masih terbuka lebar.

DAFTAR PUSTAKA

[1] S. K. Sownya, P. Deepika and J. Naren, ""Layers of Cloud – IaaS, PaaS and SaaS: A Survey"," pp. 1-5, 2014.

[2] S. Haselbock and R. Weinreich, "Decision guidance models for microservice monitoring," in *Proc. IEEE ICSAW*, Gothenburg, Sweden, 2017, pp. 54-61.

[3] S. Haselbock, R. Weinreich and G. Buchgeher, "An Expert Interview Study on Areas of Microservice Design," in *Proc. IEEE 11th SOCA*, Paris, France, 2018, pp. 137-144.

[4] J. Soldani, D. A. Tamburri and W. J. Van Den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," *Journal of Systems and Software,* vol. 146, pp. 215-232, 2018.

[5] M. Yang and M. Huang, "An microservices-based openstack monitoring tool," in *Proc. IEEE 10th ICSESS*, Beijing, China, 2019, pp. 706-709.

[6] J. Thalheim, A. Rodrigues, I. E. Akkus, P. Bhatotia, R. Chen, B. Viswanath and L. Jiao, "Sieve: Actionable insights from monitored metrics in distributed systems," in *Proc. 18 IFIP*, 2017, pp. 14-27.

[7] E. Fadda, P. Plebani and M. Vitali, "Optimizing Monitorability of Multi-cloud Applications," CAiSE, 2016.

[8] C. Canali and R. Lancellotti, "An adaptive technique to model virtual machine behavior for scalable cloud monitoring," in *Proc. ISCC*, Funchal, Portugal, 2014.

[9] A. Noor, D. N. Jha, K. Mitra, P. P. Jayaraman, A. Souza, R. Ranjan and S. Dustdar, "A framework for monitoring microservice-oriented cloud applications in heterogeneous virtualization environments," in *Proc. IEEE 12th CLOUD*, Milan, Italy, 2019, pp. 156-163.

[10] M. J. Kargar and A. Hanifizade, "Automation of regression test in microservice architecture," in *Proc. 4th ICWR*, Tehran, Iran, 2018, pp. 133-137.

[11] F. Pina, J. Correia, R. Filipe, F. Araujo and J. Cardroom, "Nonintrusive monitoring of microservice-based systems," in *Proc. IEEE 17th NCA*, Cambridge, MA, USA, 2018.

[12] M. Cinque, R. D. Corte and A. Pecchia, "Advancing monitoring in microservices systems," in *Proc. IEEE ISSREW*, Berlin, Germany, 2019, pp. 122-123.

[13] P. Gkikopoulos, "Data distribution and exploitation in a global microservice artefact observatory," in *Proc. IEEE World Cong. On Serv.*, Milan, Italy, 2019, pp. 319-322.

[14] G. Toffetti, S. Brunner, M. Blochlinger, F. Dudouet and A. Edmonds, "An architecture for self-managing microservices," in *Proc. Int. Work. On Aut. Inc. Man. in Cl.*, 2015, pp. 19-24.