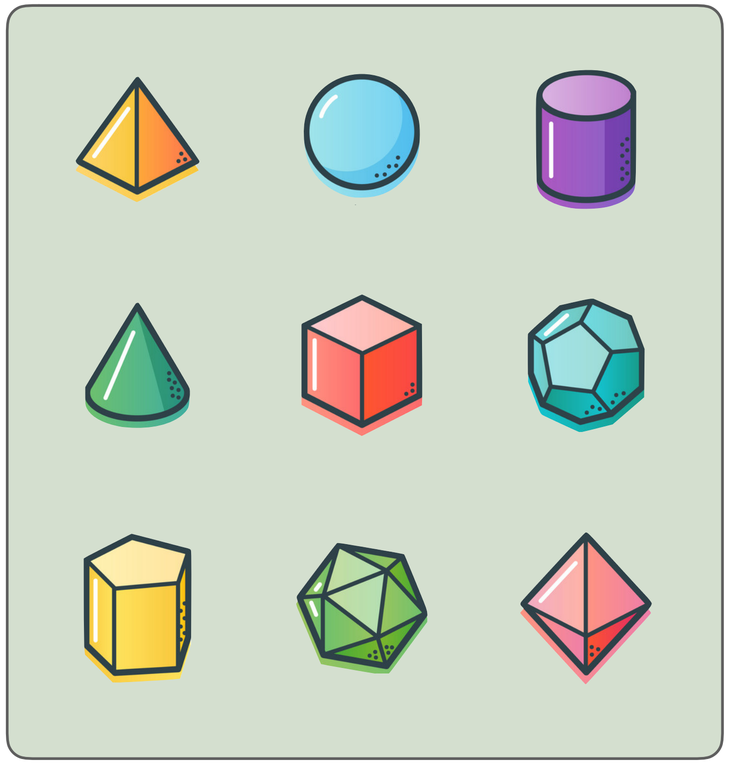
The Servers of Serverless Computing

A Formal Revisitation of Functions as Services

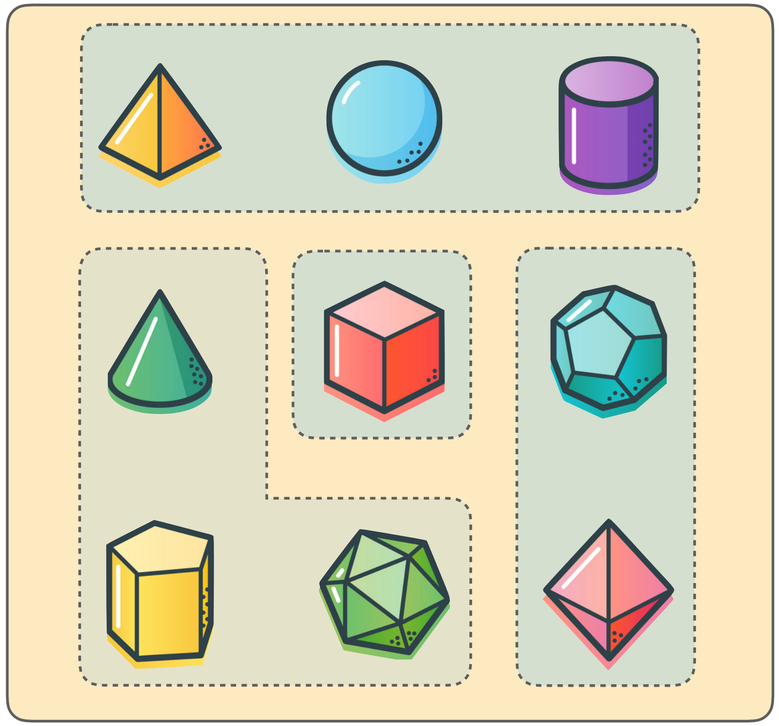
Saverio Giallorenzo[1,2*former*], Ivan Lanese[1], Fabrizio Montesi[2], Davide Sangiorgi[1], and Stefano Pio Zingaro[1]

[1]Università di Bologna/INRIA
[2]University of Southern Denmark

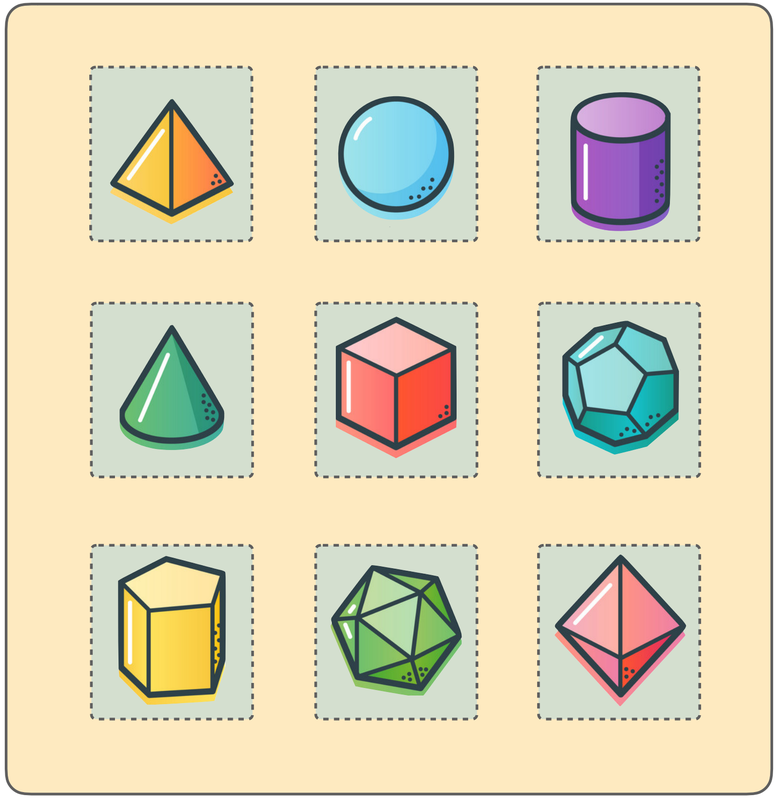
A Gentle Introduction to Serverless



Monolith



Microservices



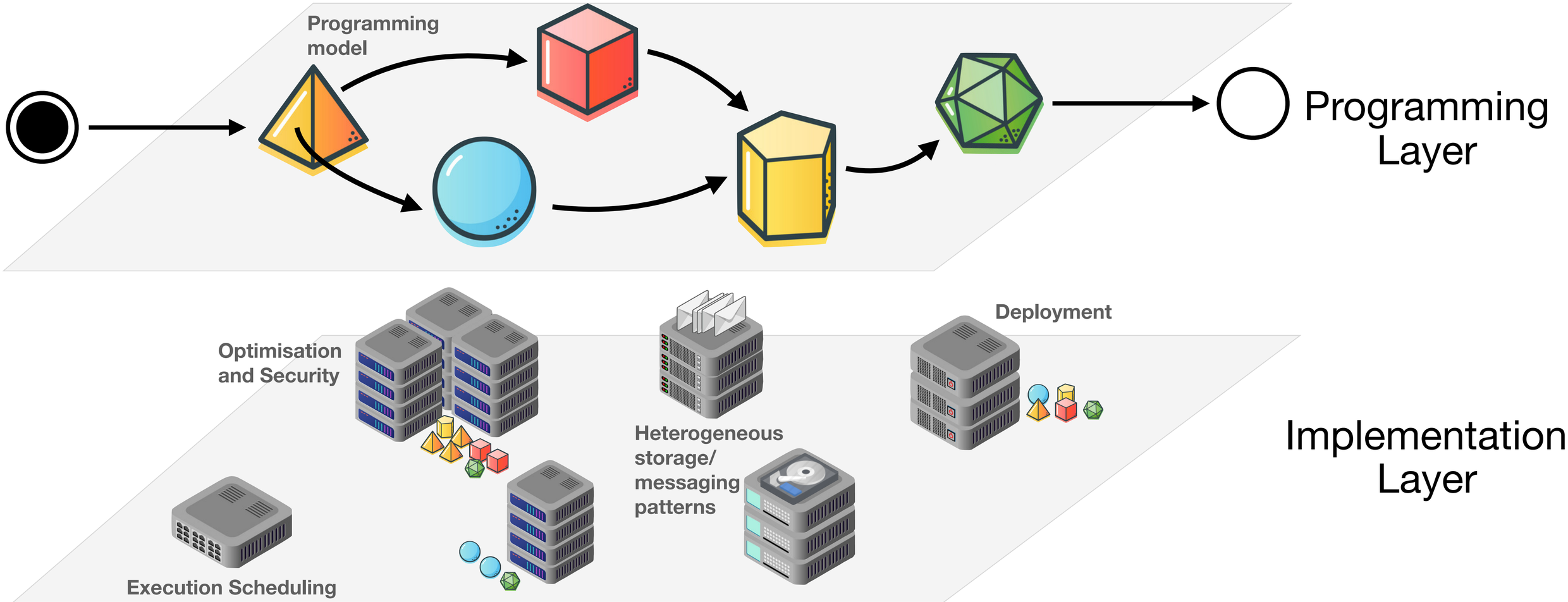
Serverless

Function

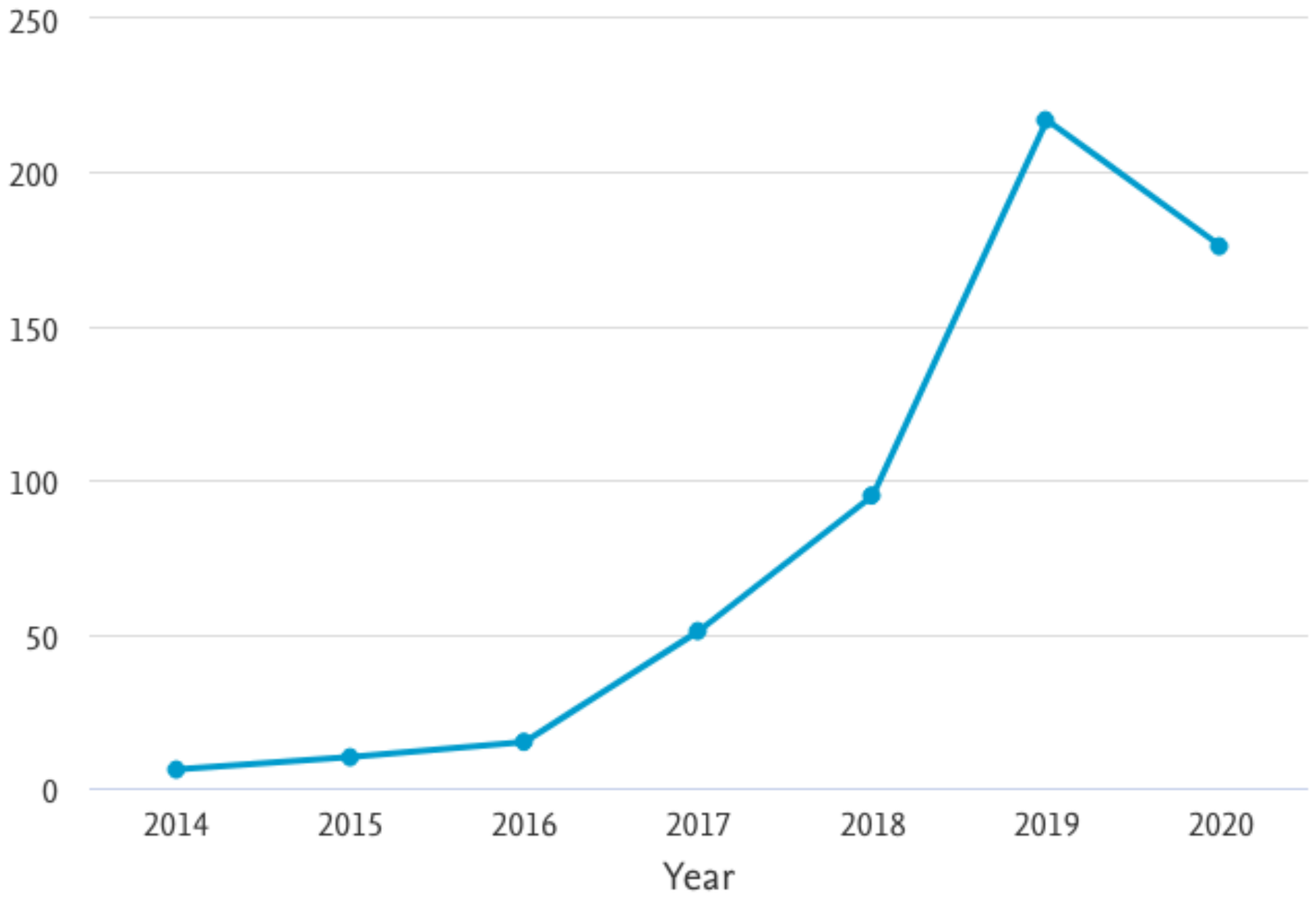
Software Unit

Runtime Environment

A Gentle Introduction to Serverless



Serverless as Research Topic



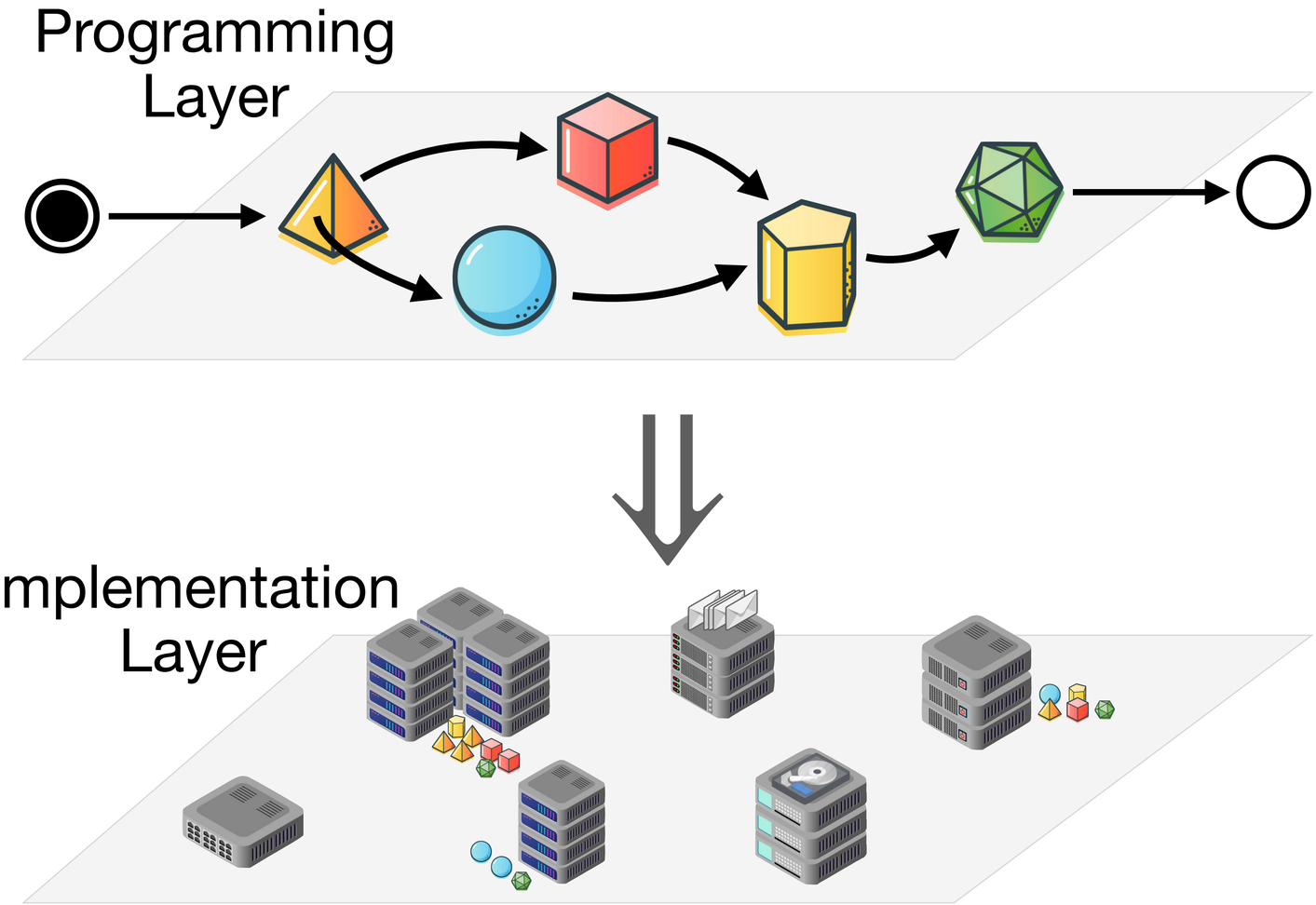
Source: Scopus

Serverless as Research Topic

Venue	# Papers	Core / SCIMAGO Rank
Future Generation Computer Systems	8	Software : Q1
IEEE Internet Computing	3	Computer Networks and Communications : Q1
IEEE Transactions on Parallel and Distributed Systems	2	Computational Theory and Mathematics: Q1
USENIX Annual Technical Conference + HotCloud	13 (6,7)	A / -
IC2E + IEEE CLOUD + CLOSER	20 (5,10,5)	- / B / -
ACM Symposium on Cloud Computing (SoCC)	12	-
SIGMOD	4	A*
Middleware	4	A
CIDR	3	A
OOPSLA	2	A*
ICSE	2	A*
INFOCOM	2	A*

Source: DBLP

The Servers of Serverless



SKC

influenced by λ and π calculus

π calculus

SKC · Simple Example (async)

$$\langle c \blacktriangleleft \mathsf{async}\ \mathsf{call}\ f, \mathcal{D} \rangle$$

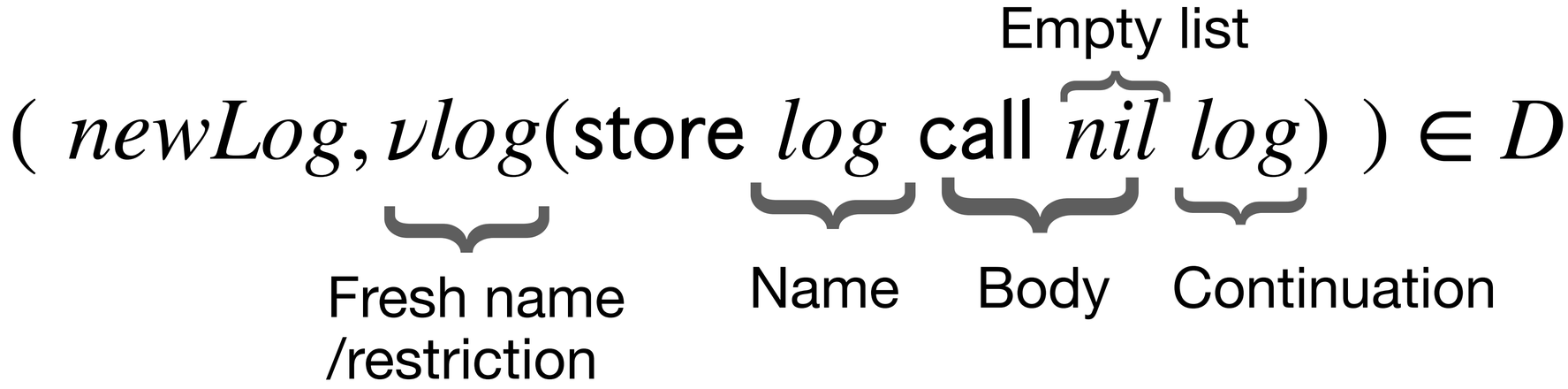
$$\longrightarrow \langle \nu c' \, (c \blacktriangleleft c' \mid c' \blacktriangleleft \mathsf{call}\ f), \mathcal{D} \rangle$$

$$\longrightarrow \langle \nu c' \, (c \blacktriangleleft c' \mid c' \blacktriangleleft M), \mathcal{D} \rangle$$

$$\longrightarrow \langle \nu c' \, (c \blacktriangleleft c' \mid c' \blacktriangleleft V), \mathcal{D} \rangle$$

$$\longrightarrow \langle \nu c' \, (c \blacktriangleleft V \mid c' \blacktriangleleft V), \mathcal{D} \rangle$$

SKC · Example, Private State



SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil \ log) \ ) \in D$$

SKC · Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil \ log) \ ) \in D$$

$$\langle c \blacktriangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ \text{call } newLog, D \rangle$$

SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil \ log) \ ) \in D$$

$$\langle c \triangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ \text{call } newLog, D \rangle$$

$$\langle c \triangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ \nu log(\text{store } log \text{ call } nil \ log), D \rangle$$

SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil \ log) \ ) \in D$$

$$\langle c \triangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ \text{call } newLog, D \rangle$$

$$\langle c \triangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ \nu log(\text{store } log \text{ call } nil \ log), D \rangle$$

$$\nu log \langle c \triangleleft (\lambda x \,.\, (\text{call } pair \ ((M \ x)(N \ x)) \ x)) \ log, D \cup \{(log, \text{call } nil)\} \rangle$$

SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil\ log)\ ) \in D$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \text{call } newLog, D\rangle$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \nu log(\text{store } log \text{ call } nil\ log), D\rangle$$

$$\nu log\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ log, D \cup \{(log, \text{call } nil)\}\rangle$$

$$\nu log\langle c \triangleleft \text{call } pair\ ((M\ log)(N\ log))\ log, D \cup \{(log, \text{call } nil)\}\rangle$$

SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil\ log)\ ) \in D$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \text{call } newLog, D \rangle$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \nu log(\text{store } log \text{ call } nil\ log), D \rangle$$

$$\nu log \langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ log, D \cup \{(log, \text{call } nil)\} \rangle$$

$$\nu log \langle c \triangleleft \text{call } pair\ ((M\ log)(N\ log))\ log, D \cup \{(log, \text{call } nil)\} \rangle$$

$$\nu log \langle c \triangleleft \text{call } pair\ (M\ log\ V_N)\ log, D \cup \{(log, N_{log})\} \rangle$$

SKC • Example, Private State

$$(\ newLog, \nu log(\text{store } log \text{ call } nil\ log)\ ) \in D$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \text{call } newLog, D \rangle$$

$$\langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ \nu log(\text{store } log \text{ call } nil\ log), D \rangle$$

$$\nu log \langle c \triangleleft (\lambda x\,.\,(\text{call } pair\ ((M\ x)(N\ x))\ x))\ log, D \cup \{(log, \text{call } nil)\} \rangle$$

$$\nu log \langle c \triangleleft \text{call } pair\ ((M\ log)(N\ log))\ log, D \cup \{(log, \text{call } nil)\} \rangle$$

$$\nu log \langle c \triangleleft \text{call } pair\ (M\ log\ V_N)\ log, D \cup \{(log, N_{log})\} \rangle$$

$$\nu log \langle c \triangleleft \text{call } pair\ V_M\ log, D \cup \{(log, N_{log} :: M_{log})\} \rangle$$

SKC • Results, $SKC \leftrightarrow \pi$ Operational Correspondence

Theorem 1. From SKC-to-π operational correspondence

If $C \rightarrow C'$ then $[\![C]\!]^* \rightarrow \approx [\![C']\!]^*$

Theorem 2. From π-to-SKC operational correspondence.

If $\{[C]\}\ ^* \rightarrow P$ then there is C' with $C \rightarrow C'$ and $P \approx [\![C']\!]^*$

SKC • Future Work

- **guarantees** like *sequential execution/consistency* (global total order) and weaker forms, like *global-state transformation serialisability* (equivalence to a global serial schedule);
- **programming models** to have a global view of the logic of the distributed functions, yet capturing the loosely-consistent execution model of Serverless (e.g., choreographies);
- **transformation frameworks**, e.g., depending on the application context and inbound load, users/optimisation systems can transform parts of a given system from Serverless to Microservices and vice versa;
- **prediction models** for cost/resource usage, which require a modelling that relates functions and their execution at the implementation layer.

Thank for your time

Happy cruising!