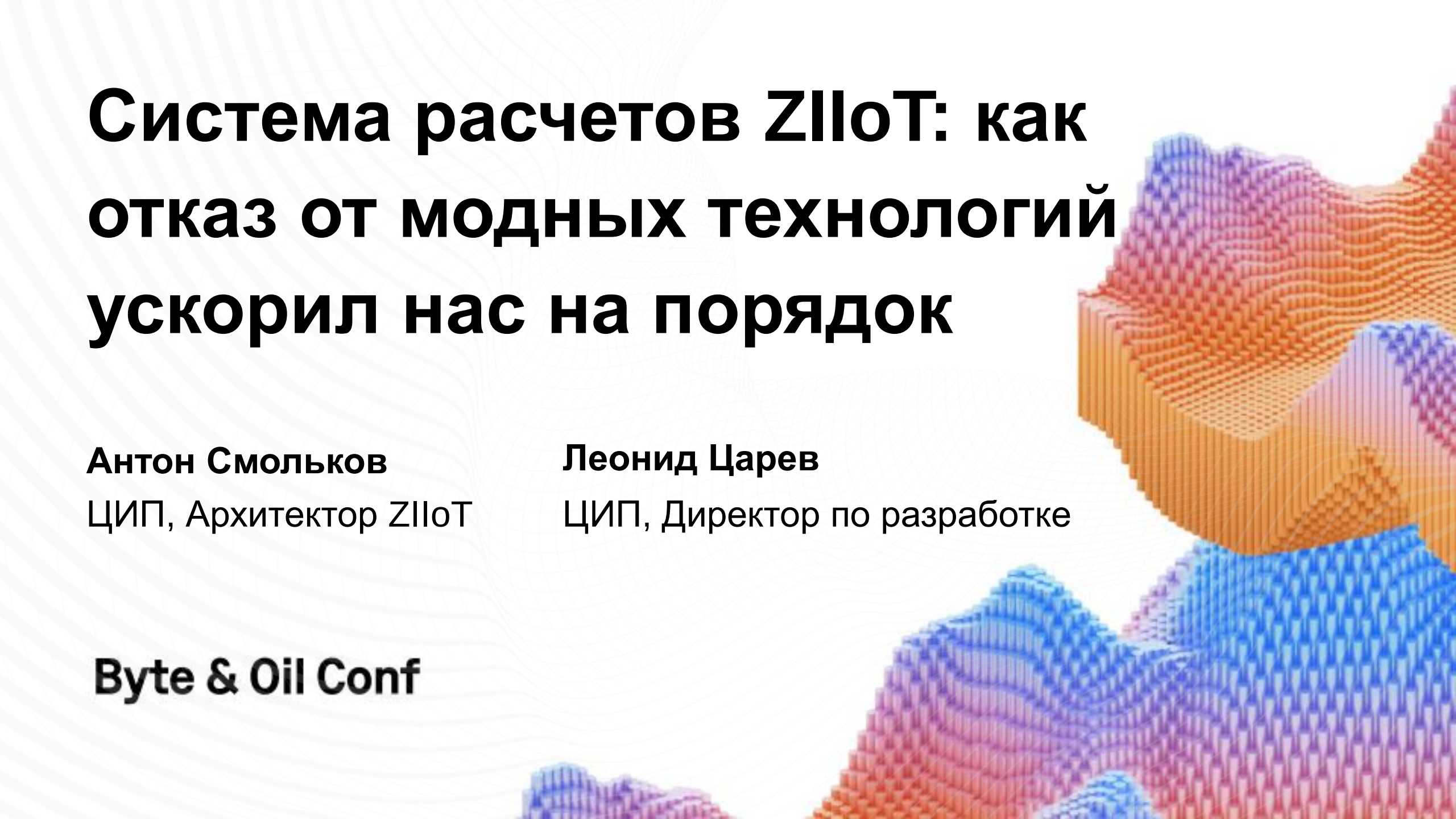
Система расчетов ZIIoT: как отказ от модных технологий ускорил нас на порядок

Антон Смольков
ЦИП, Архитектор ZIIoT

Леонид Царев
ЦИП, Директор по разработке



Byte & Oil Conf

ЦИФРОВАЯ
ИНДУСТРИАЛЬНАЯ
ПЛАТФОРМА

Система расчетов ZIIoT: как отказ от модных технологий ускорил нас на порядок

idp.zyfra.com

Спикеры

Антон Смольков
Архитектор платформы ZIIoT
Области интересов — инфраструктурные сервисы, базисное ПО, производительность.
До этого в качестве .NET-разработчика работал над расчетами ZIIoT, а еще ранее был начальником отдела DevOps.

✉ anton.smolkov@idpllc.ru

Леонид Царев
Директор по разработке ООО «ЦИП»
Отвечает за разработку продуктов компании, в том числе платформы ZIIoT.
До прихода в ЦИП Леонид был архитектором и зам. начальника отдела разработки в ГК Монополия, где запустил monopoly.online.

✉ leonid.tsarev@idpllc.ru

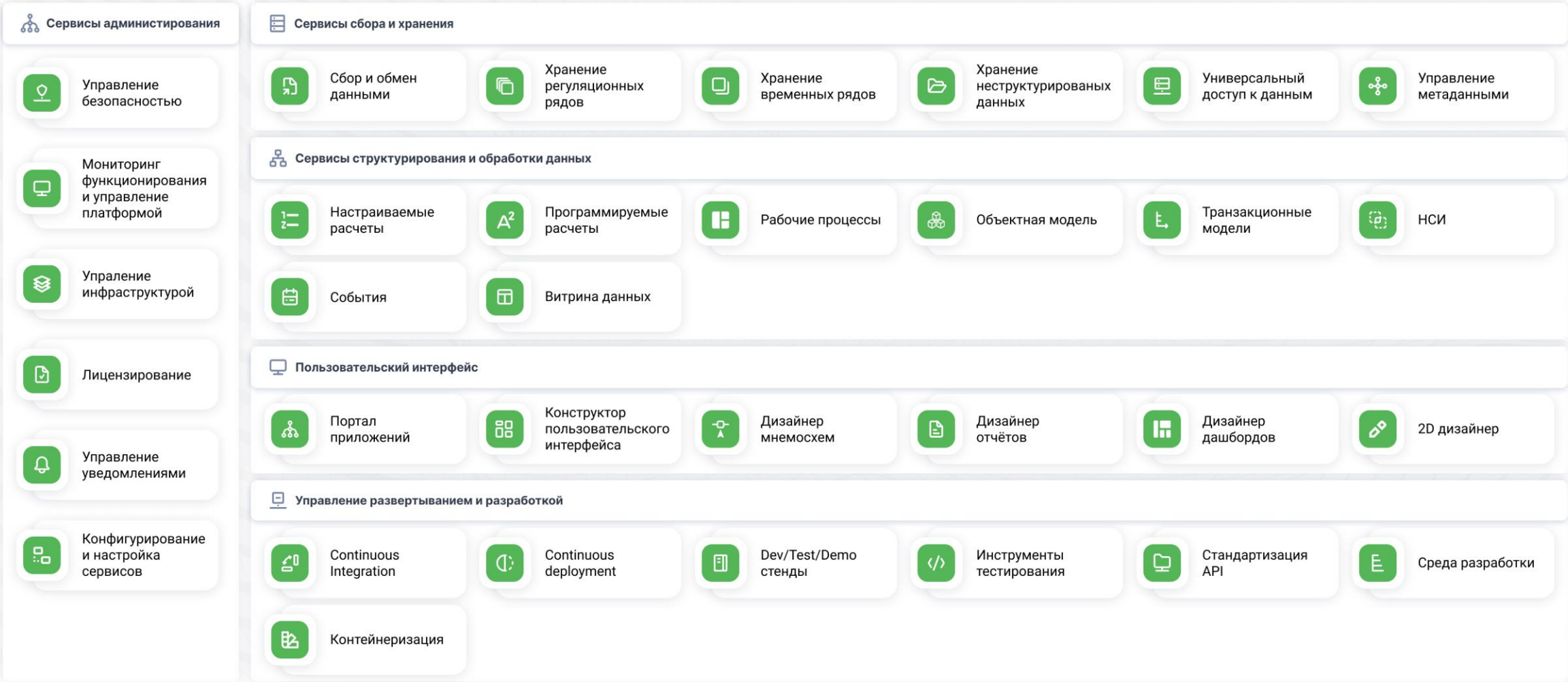
О чем будем рассказывать

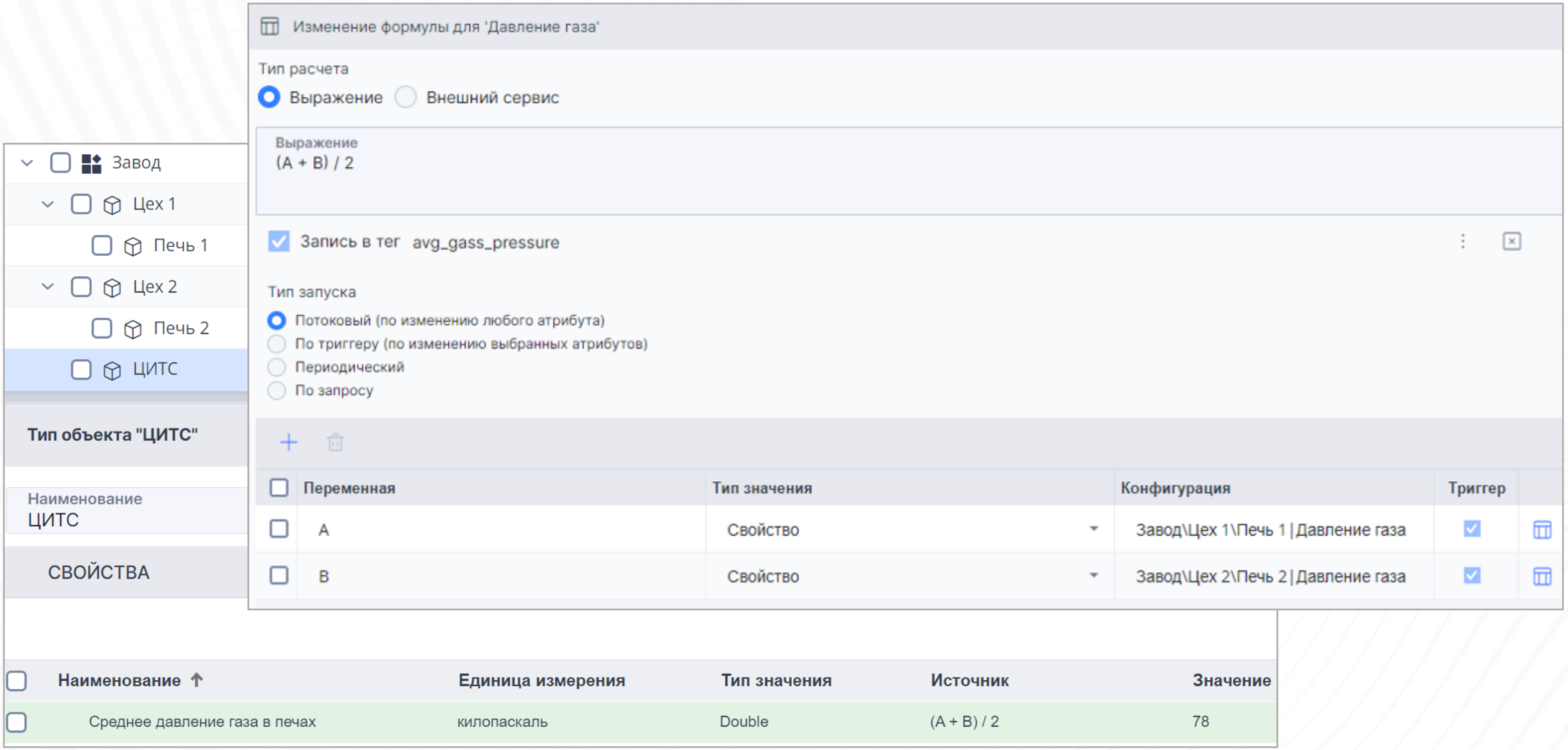
- ✓ Контекст, платформа ZIIoT
- ✓ Модуль расчётов, первоначальное решение, металлургический завод
- ✓ Внедрение на НПЗ и как там всё не работало
- ✓ Рефакторинг архитектуры
- ✓ Результаты и выводы

Контекст

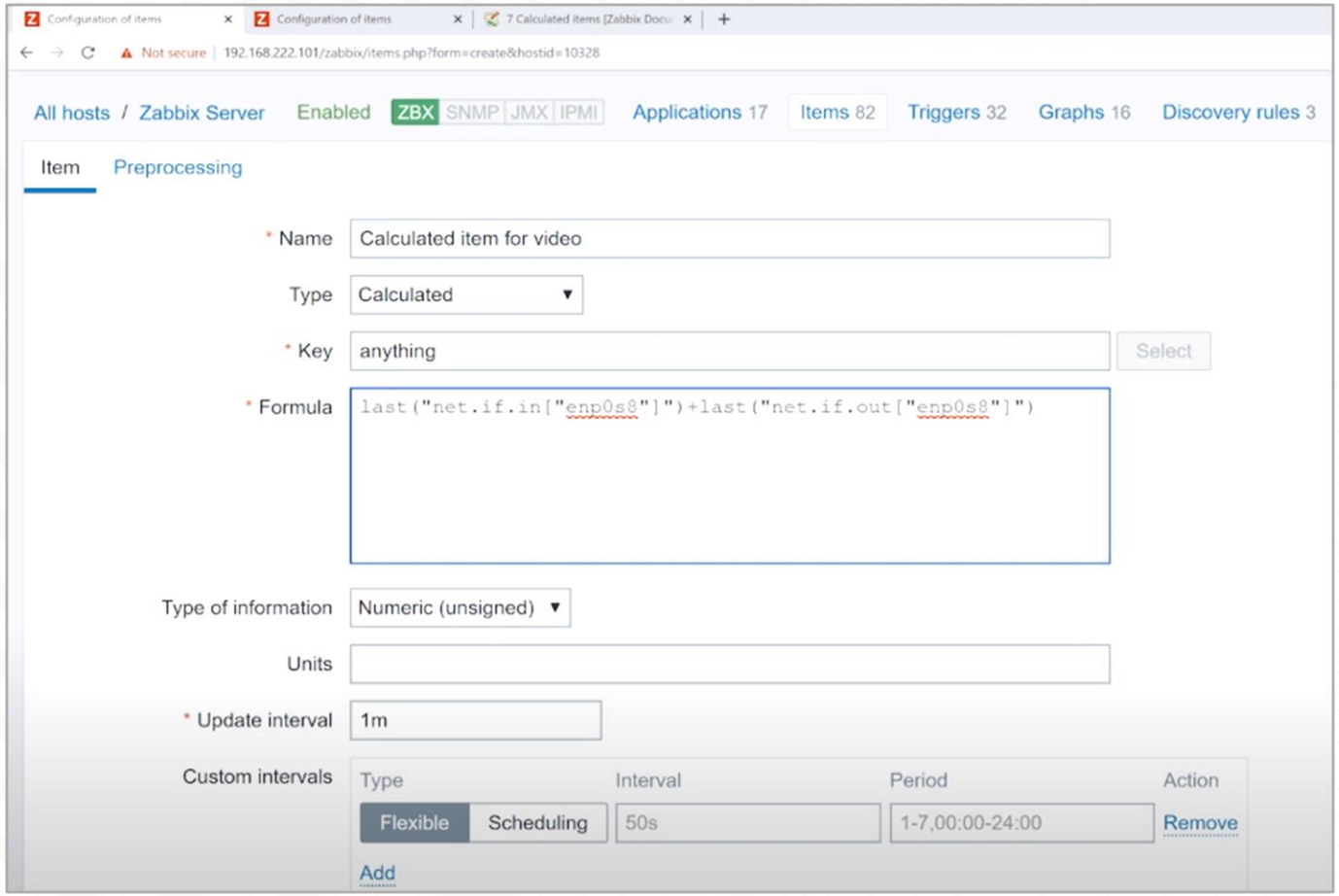
АСУТП, MES, ERP и место платформы в этом

Zyfra Industrial IoT
Функциональная архитектура

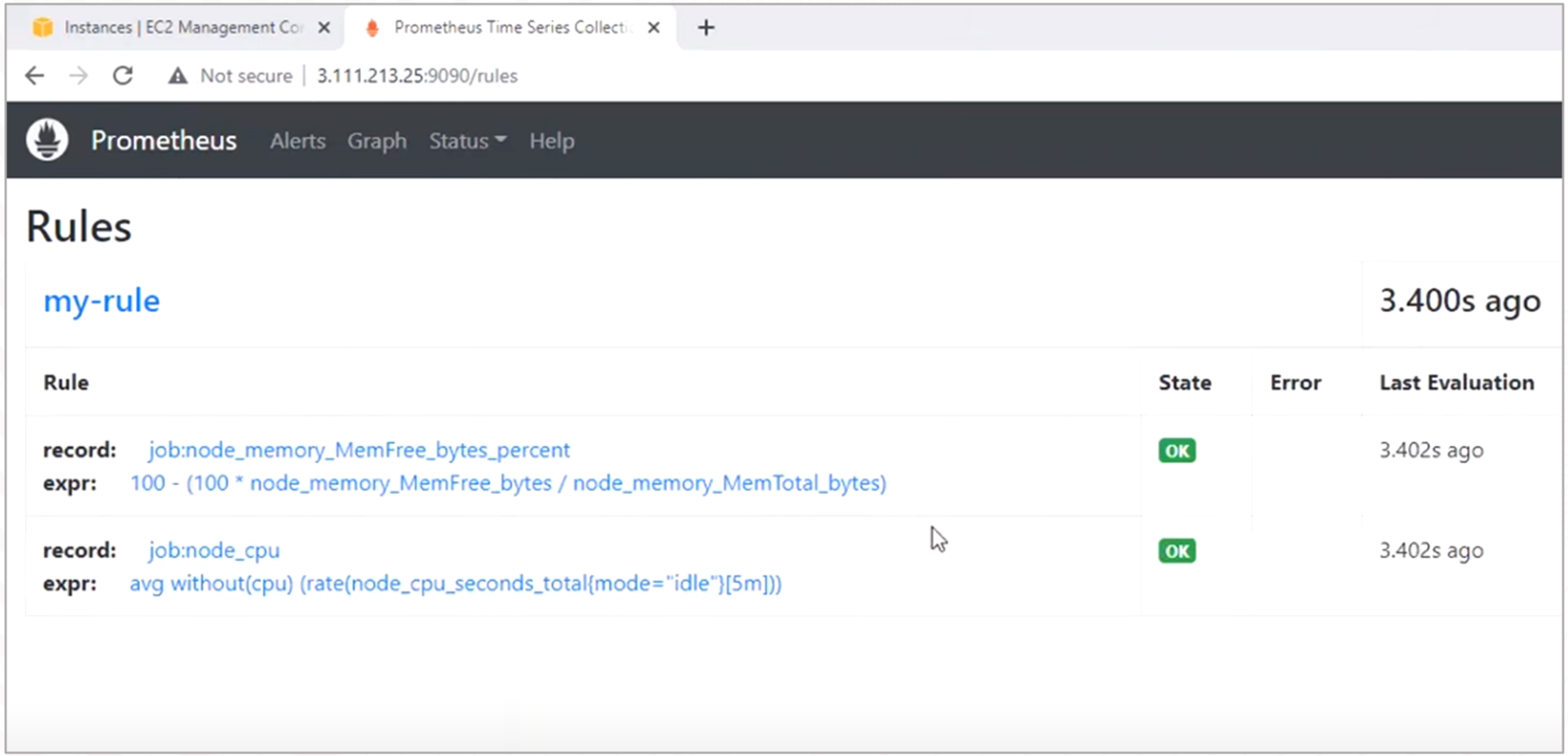



Изменение формулы для 'Давление газа'
Тип расчета
Выражение
Внешний сервис
Выражение
(A + B) / 2
Запись в тег avg_gass_pressure
Тип запуска
Потоковый (по изменению любого атрибута)
По триггеру (по изменению выбранных атрибутов)
Периодический
По запросу
Переменная
Тип значения
Конфигурация
Триггер
A
Свойство
Завод\Цех 1\Печь 1 | Давление газа
B
Свойство
Завод\Цех 2\Печь 2 | Давление газа
Завод
Цех 1
Печь 1
Цех 2
Печь 2
ЦИТС
Тип объекта "ЦИТС"
Наименование
ЦИТС
СВОЙСТВА
Наименование
Единица измерения
Тип значения
Источник
Значение
Среднее давление газа в печах
килопаскаль
Double
(A + B) / 2
78

Аналогии: Zabbix calculated items



Аналогии: Prometheus record rules

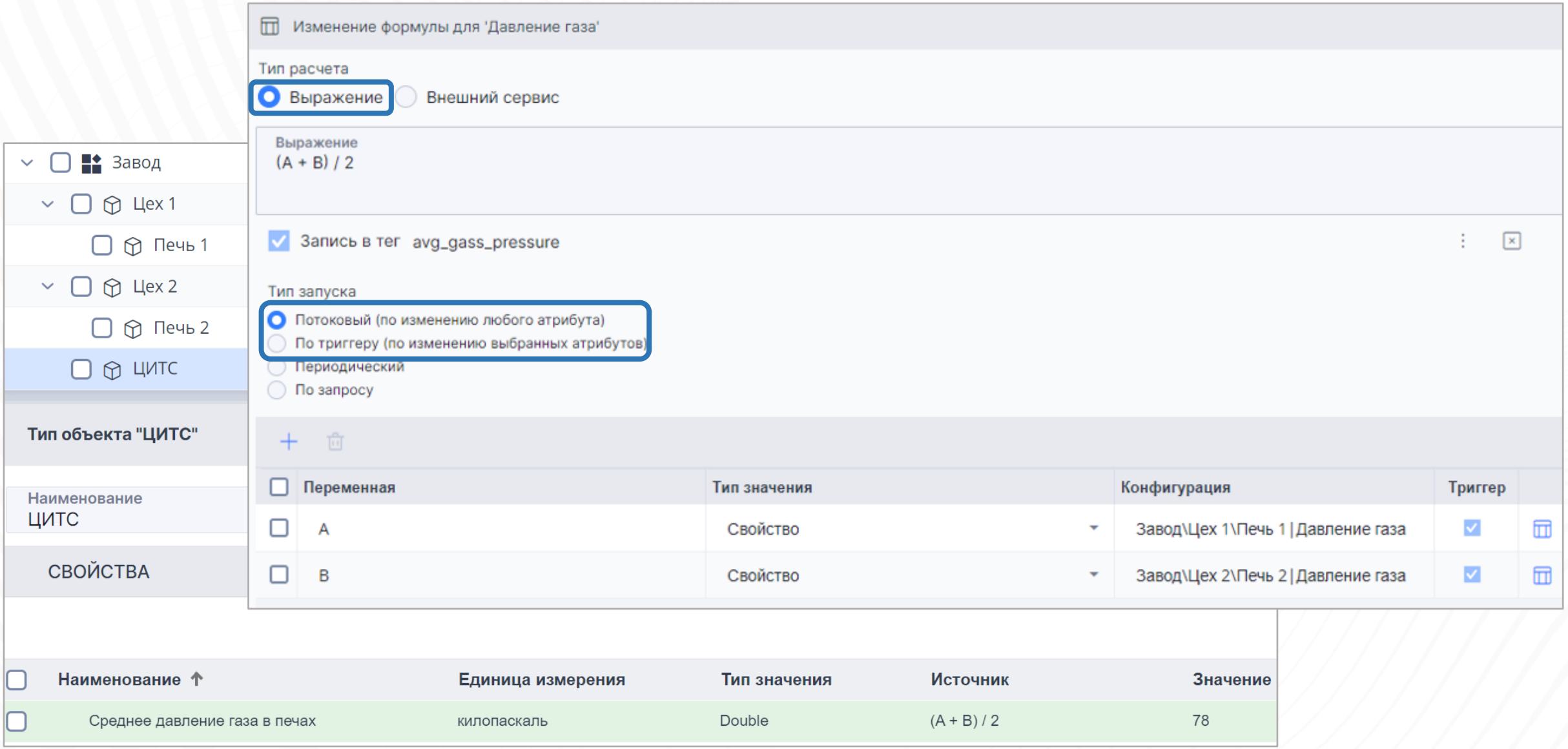


Первоначальное решение

Требования от металлургического завода

Функциональные

Тип расчёта	MVEL-выражение
Способ запуска	По подписке на значения (потоковый/по триггеру)


Изменение формулы для 'Давление газа'
Тип расчета
Выражение
Внешний сервис
Выражение
(A + B) / 2
Запись в тег avg_gass_pressure
Тип запуска
Потоковый (по изменению любого атрибута)
По триггеру (по изменению выбранных атрибутов)
Периодический
По запросу
Переменная
Тип значения
Конфигурация
Триггер
A
Свойство
Завод\Цех 1\Печь 1 | Давление газа
B
Свойство
Завод\Цех 2\Печь 2 | Давление газа
Завод
Цех 1
Печь 1
Цех 2
Печь 2
ЦИТС
Тип объекта "ЦИТС"
Наименование
ЦИТС
СВОЙСТВА
Наименование
Единица измерения
Тип значения
Источник
Значение
Среднее давление газа в печах
килопаскаль
Double
(A + B) / 2
78

Язык MVEL

MVFLEX Expression Language (MVEL) — встраиваемый язык выражений для платформы JVM. Обычно используется для предоставления возможности описания базовой логики конечным пользователям через средства конфигурирования.

MVEL — это урезанная Java:

- sum = A + B +C
- min(A, B)
- def fn(...)
- object.property
- if (A+B>12)
- *System.out.println("Hello, world!");*
- *(объявлять свои классы нельзя)*

```
Vp=Vpsource*Vpmult;
P=Psource*Pmult;
dP=Perepad*Math.pow(Vp/Shkala, 2);
Krash=1-(0.351+0.256*Math.pow(betta, 4)+0.93* Math.pow(betta, 8))*(1-Math.pow(1-dP/10000/(P+1.026), 1/X));
Qm=0.000012522*Alfa*Krash*d20*d20*K0*K0*Kp*Ksh*Math.pow(dP*Ror*283.73*(P + 1.026)/((T + 273.15)*Ksg), 0.5);
return Qm;
```

```
zifTime=Fn.bod('*+3h')
  .minusHours(3)
  .format(java.time.format.DateTimeFormatter.ofPattern('dd.MM.yyyy HH:mm:ss'));
return (A + Fn.avg($A,'y',zifTime)) *
       (B + Fn.min($B,'y',zifTime)) *
       (C + Fn.max($C,'y',zifTime));
```

Требования от металлургического завода

Функциональные

Тип расчёта	MVEL-выражение
Способ запуска	По подписке на значения (потоковый/по триггеру)

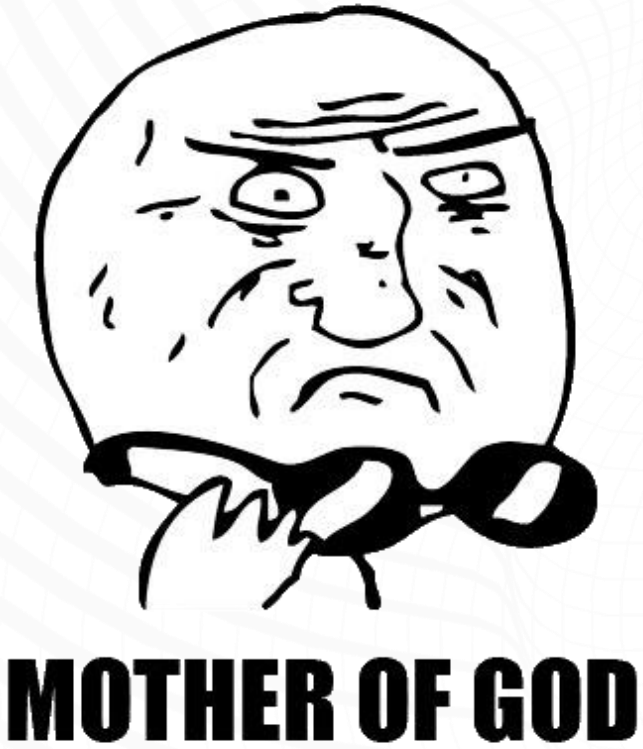
Нефункциональные

- Поддержка одновременной работы ~10_000 расчётов

Требования от металлургического завода

То есть нужно 10к штуковин, которые:

1. Подпишутся на очередь с потоком данных
2. При поступлении данных будут что-то вычислять по своим формулам и записывать результат
3. Будут иметь состояние



Да это же акторы!

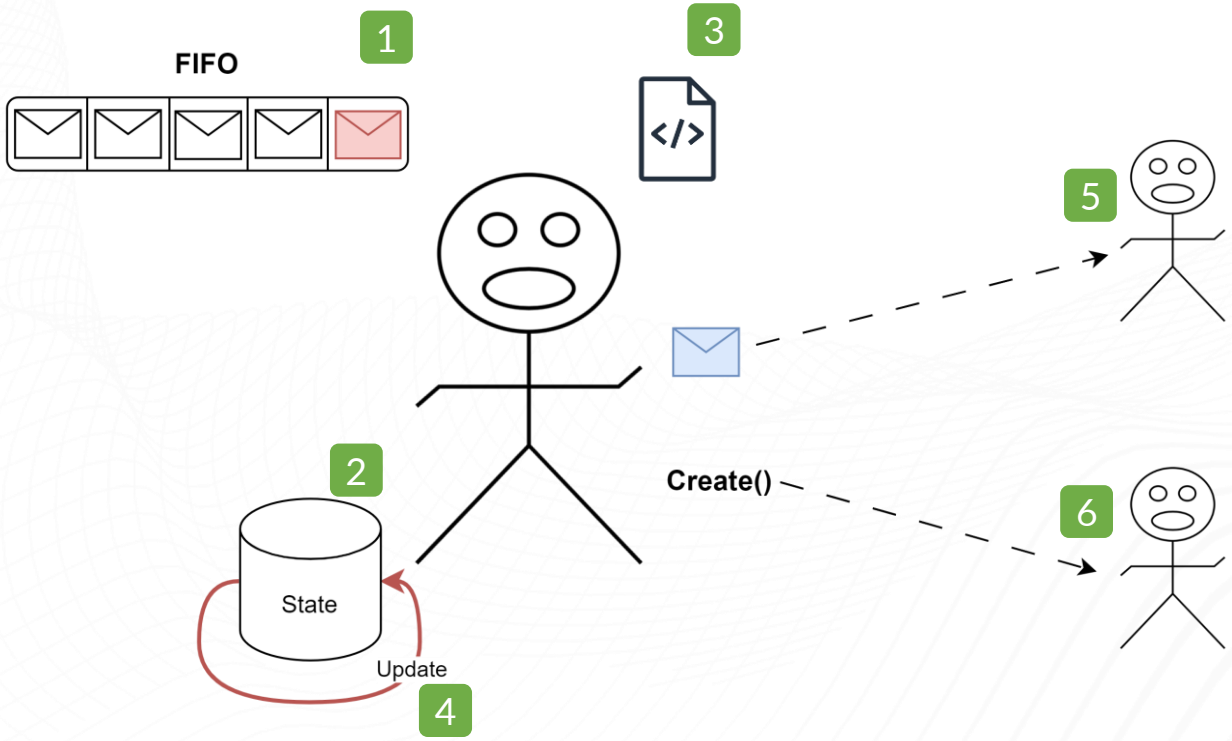
Актор – базовый однопоточный строительный блок системы.
Наносервис внутри приложения.

Имеет:

1. очередь сообщений
2. состояние

Получив сообщение, может:

3. выполнить какую-то работу
4. изменить своё состояние *(в т.ч. тем самым изменив способ обработки последующих сообщений)*
5. отправить сообщение другому актору
6. создать дочернего актора



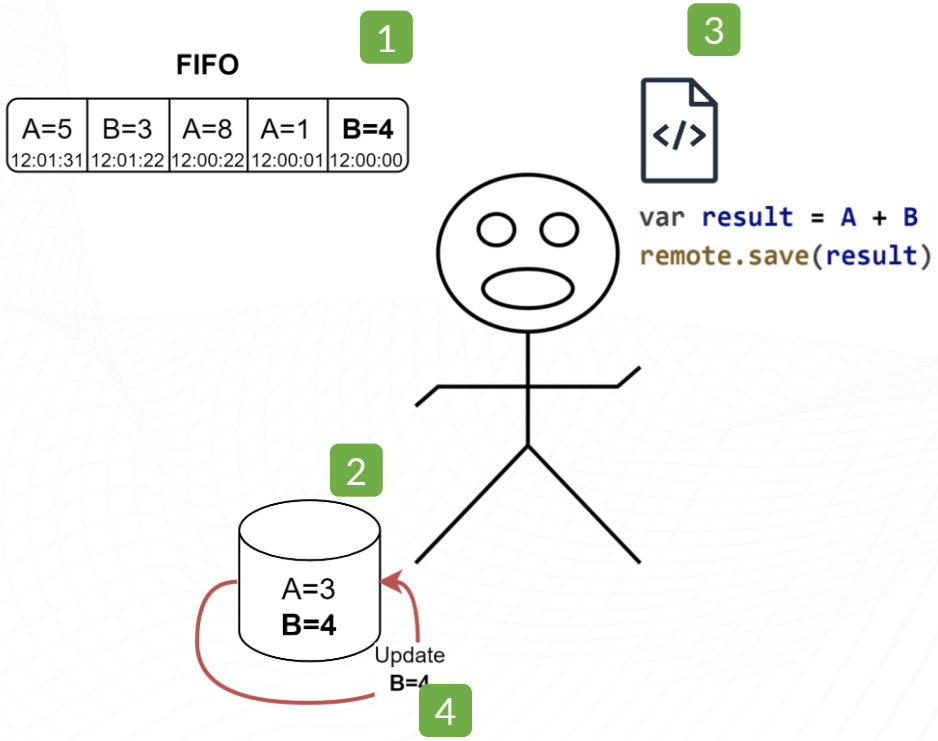
Расчёт – частный случай актора

Имеет:

1. очередь сообщений
2. состояние

Получив сообщение, может:

3. выполнить какую-то работу
4. изменить своё состояние *(в т.ч. тем самым изменив способ обработки последующих сообщений)*

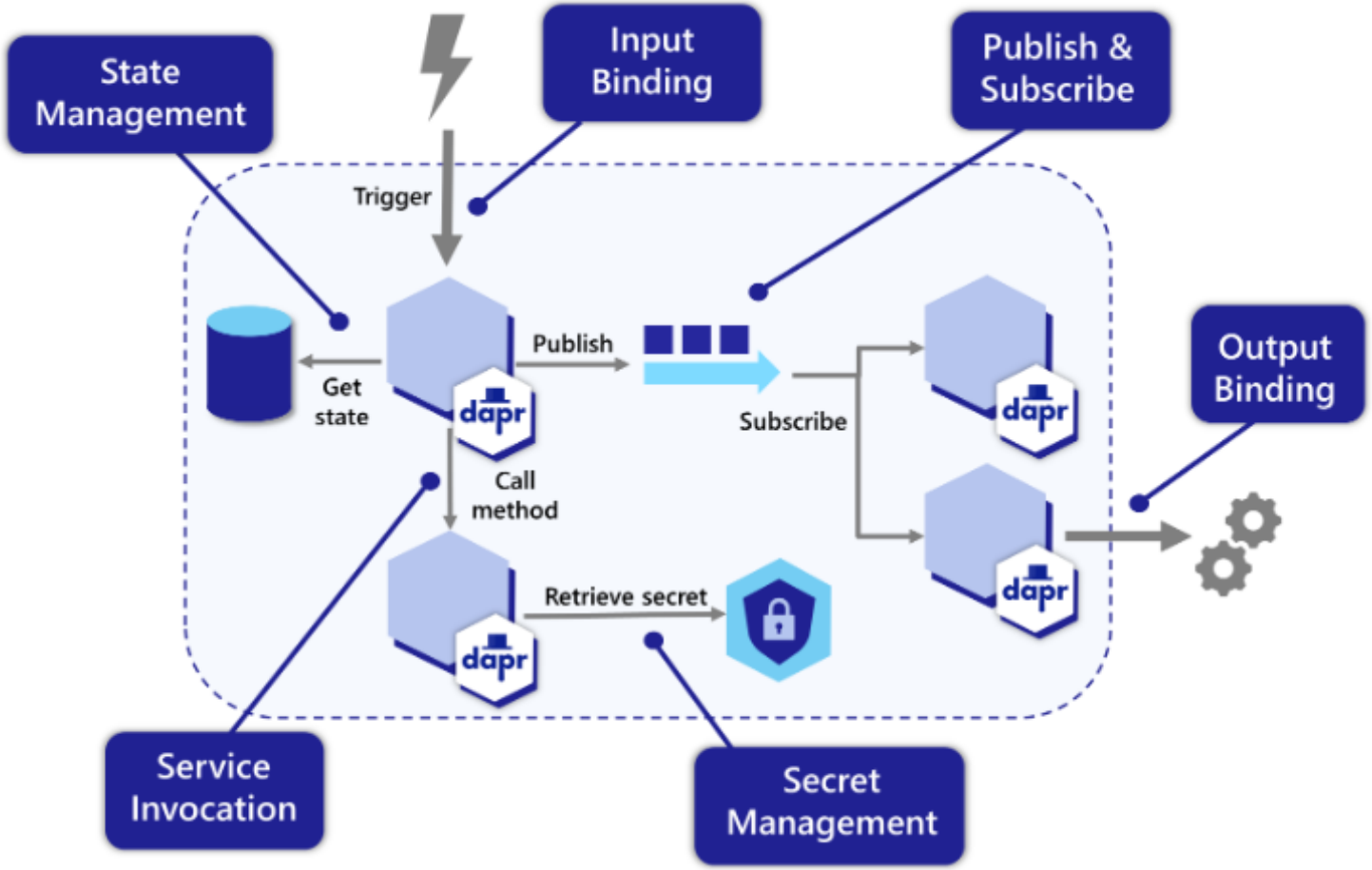


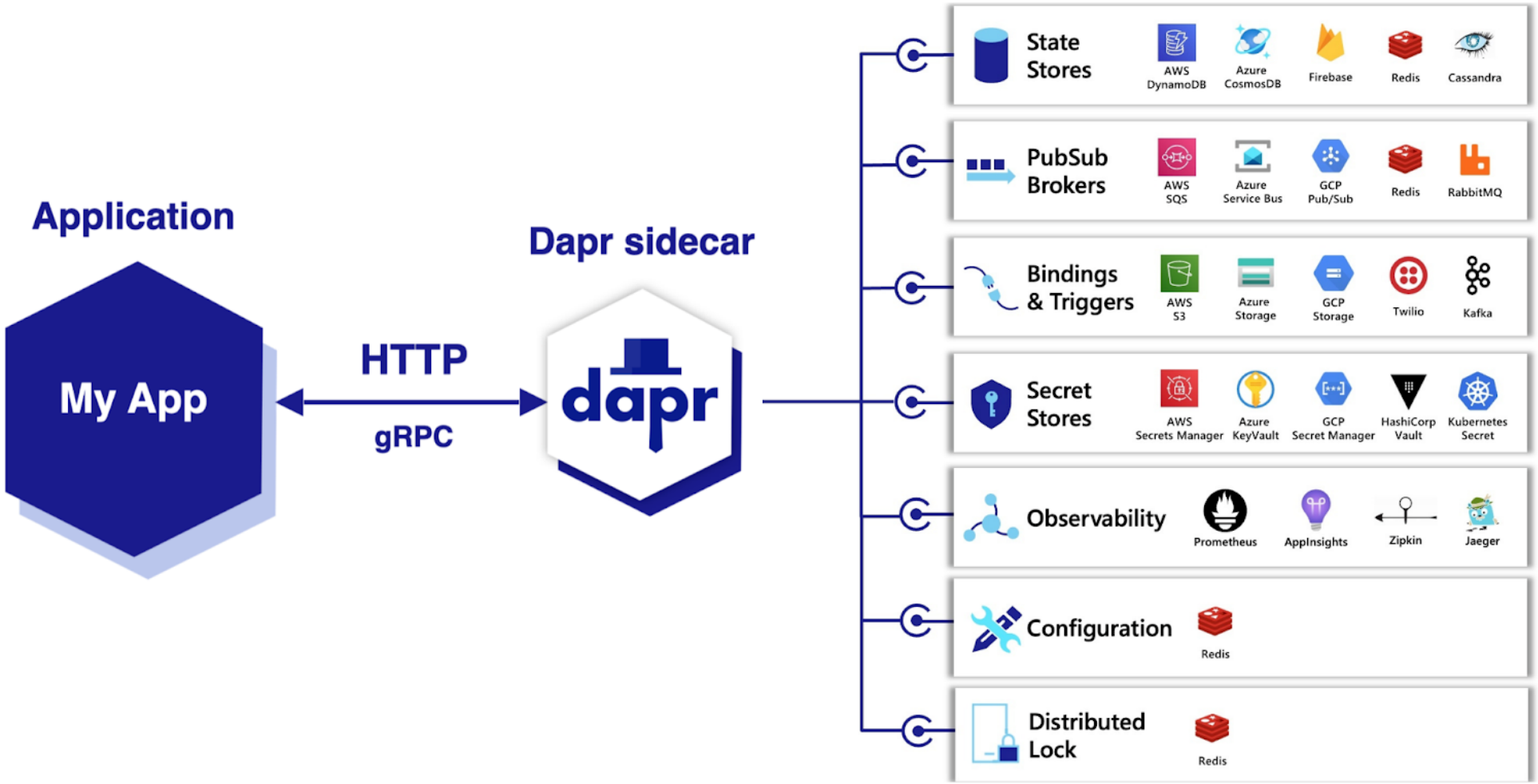
Выбор акторного фреймворка

Критерий	Akka.Net	Orleans	Proto.Actor	Dapr
Мультиплатформенность (Java/.Net)	✗	✗	✗	✓
Виртуальные акторы	✗	✓	✓	✓
Cloud native (k8s)	✗	✓	✓	✓
Авторебаланс при масштабировании	✓	✗*	✓	✓
Big tech за плечами	✗	✓	✗	✓
Зрелость	✓	✓	✗	✗
Доп. полезности	✗	✗	✗	✓

Фреймворк Dapr — суперкомбайн, который решает все проблемы

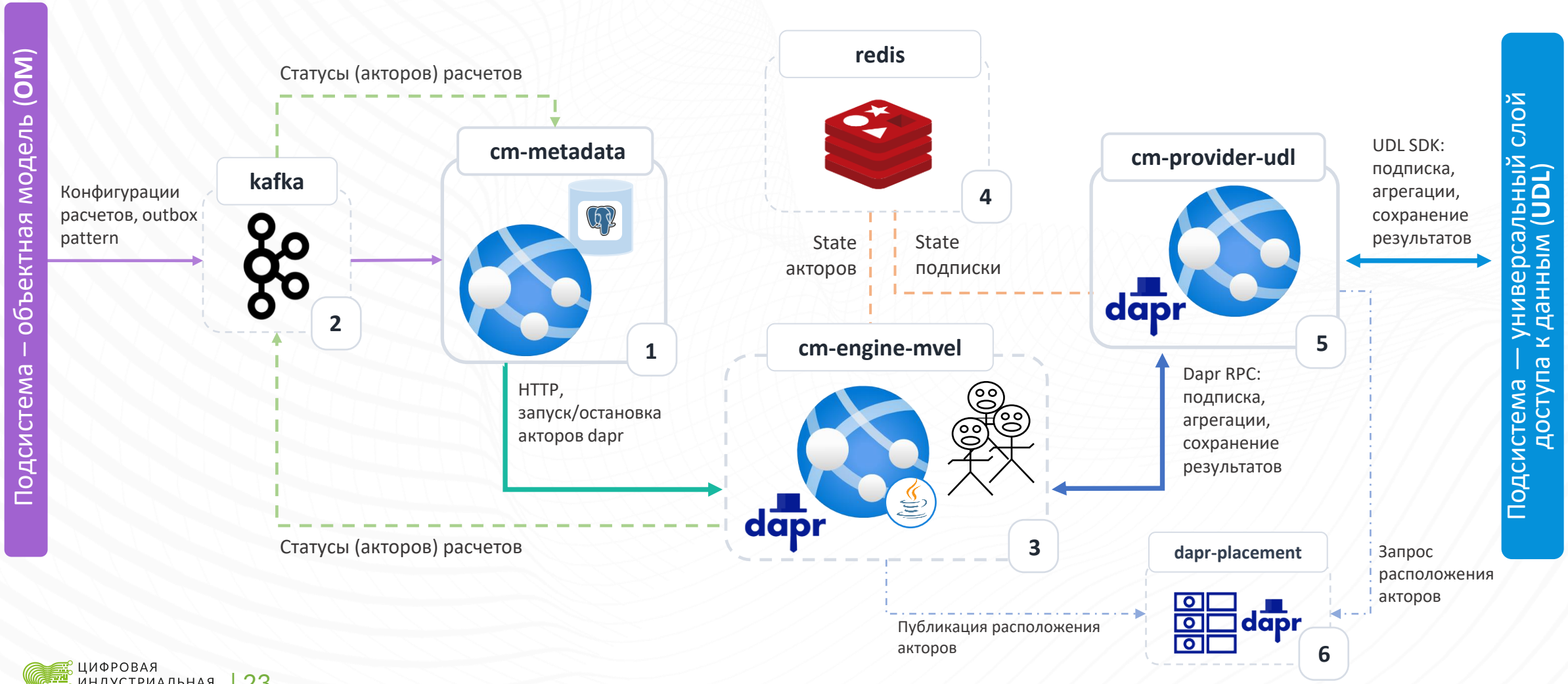
- Publish & Subscribe
- Service Invocation
- Secret Management
- Input/Output Bindings
- State Management
- Virtual Actors




Application
My App
HTTP
gRPC
Dapr sidecar
dapr
State Stores
AWS DynamoDB
Azure CosmosDB
Firebase
Redis
Cassandra
PubSub Brokers
AWS SQS
Azure Service Bus
GCP Pub/Sub
Redis
RabbitMQ
Bindings & Triggers
AWS S3
Azure Storage
GCP Storage
Twilio
Kafka
Secret Stores
AWS Secrets Manager
Azure KeyVault
GCP Secret Manager
HashiCorp Vault
Kubernetes Secret
Observability
Prometheus
AppInsights
Zipkin
Jaeger
Configuration
Redis
Distributed Lock
Redis

Подсистема расчётов (CM)

1. **cm-metadata** – конфигурация, управление, статусы расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, статусы расчётов (акторов)
3. **cm-engine-mvel** – исполнение MVEL- расчётов, хостинг акторов расчётов (java)
4. **redis** – Dapr state store. Акторы расчётов хранят в нём свое состояние
5. **cm-provider-udl** – сервис-фасад для UDL. Тоже использует Dapr state-store
6. **dapr-placement** – “DNS” для акторов Dapr



ТЭЦ. Паровоздуходувная станция

1 °C

Данные за период с 10.02.2022 13:58 по 11.02.2022 13:58

11.02.2022 13:58:03

Показатель, ед. изм.		ТВД-4	ТВД-5	ТВД-7	ТВД-8	ЭВД-9
Направление ТВД			ДП №5	ДП №4	ДП №3	ДП №5
Обороты	1/мин	3173	3358	3131		
Расход дутья	м³/мин	3806.1	5109.0	4167.9	-18.3	1059.0
Давление дутья	кгс/см²	3.2	4.1	3.7	0	4.3
Концентрация O_2	%	30.4	28.0	30.2	20.2	
Расход O_2	тыс. м³/ч	28.7	42.4	31.8	0	8.1
Z (Запас до линии настройки ППЗ)	%	8.9	9.7	9.2		
Степень открытия СНОРТ	%	0	0	0		
Рециркуляция воздуха	м³/мин	742.7	63.2	266.7		
Удельный расход пара на выработку ДД	тыс. ккал/тыс. м³	102.00	88.20	86.60	0	

Показатель	ТВД-4	ТВД-5	ТВД-7	ТВД-8	ЭВД-9
Состояние конденсатора					
Состояние воздухоохладителя					

Уд. расход теплоэнергии в паре на производство доменного дутья тыс. ккал/тыс.м³ привед.

сут: -2% | 89,0 | -2,1 / 91,1

мес: -27% | 90,2 | -34,1 / 124,4

Уд. расход электроэнергии на производство доменного дутья кВт/тыс.м³ привед.

сут: -8% | 1,329 | -0,121 / 1,450

мес: -5% | 1,382 | -0,068 / 1,450

ДП	Расход дутья на печах, м³/мин	Концентрация O_2, %
ДП-3	3 718,34	31,17
ДП-4	4 010,82	30,23
ДП-5	5 980,93	31,12

В работе
В резерве
В ремонте

Давление O_2 КЦ, кгс/см²: 0,058

Чистота O_2 КЦ, %: 98,39

Циркуляционные насосы

1 656 кВт

11 057 м³/ч

Конденсатные насосы

174 т/ч

Нефтеперерабатывающий завод, новые требования

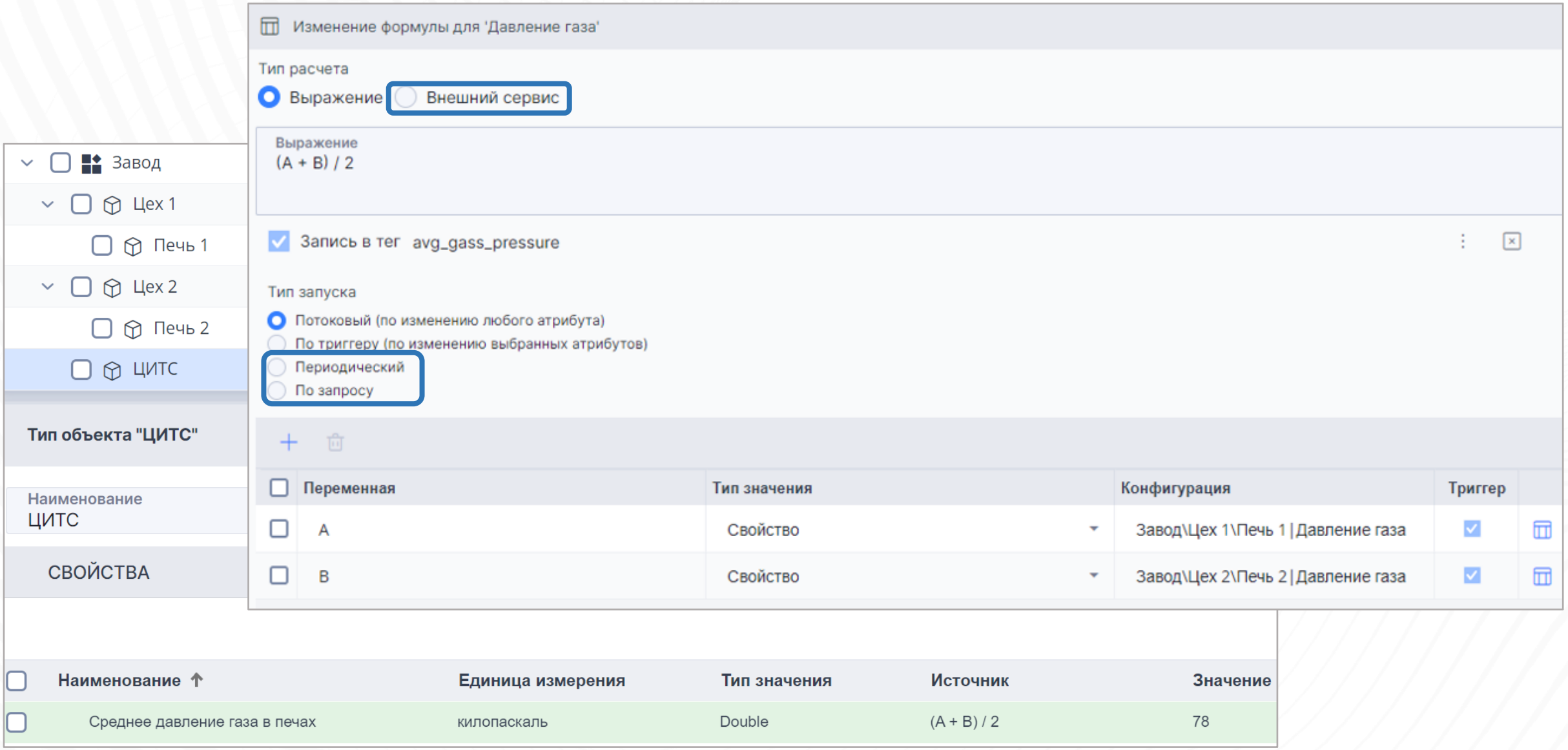
Функциональные требования

Типы расчёта:

- MVEL-выражение
- **Вызов внешнего сервиса**

Способы запуска:

- По подписке на значения (потоковый/по триггеру)
- **По таймеру (периодический)**
- **По запросу**


Изменение формулы для 'Давление газа'
Тип расчета
Выражение
Внешний сервис
Выражение
(A + B) / 2
Запись в тег avg_gass_pressure
Тип запуска
Потоковый (по изменению любого атрибута)
По триггеру (по изменению выбранных атрибутов)
Периодический
По запросу
Переменная
Тип значения
Конфигурация
Триггер
A
Свойство
Завод\Цех 1\Печь 1 | Давление газа
B
Свойство
Завод\Цех 2\Печь 2 | Давление газа
Завод
Цех 1
Печь 1
Цех 2
Печь 2
ЦИТС
Тип объекта "ЦИТС"
Наименование
ЦИТС
СВОЙСТВА
Наименование
Единица измерения
Тип значения
Источник
Значение
Среднее давление газа в печах
килопаскаль
Double
(A + B) / 2
78

Нефункциональные требования

Поддержка одновременной работы
~300_000(!) расчётов.

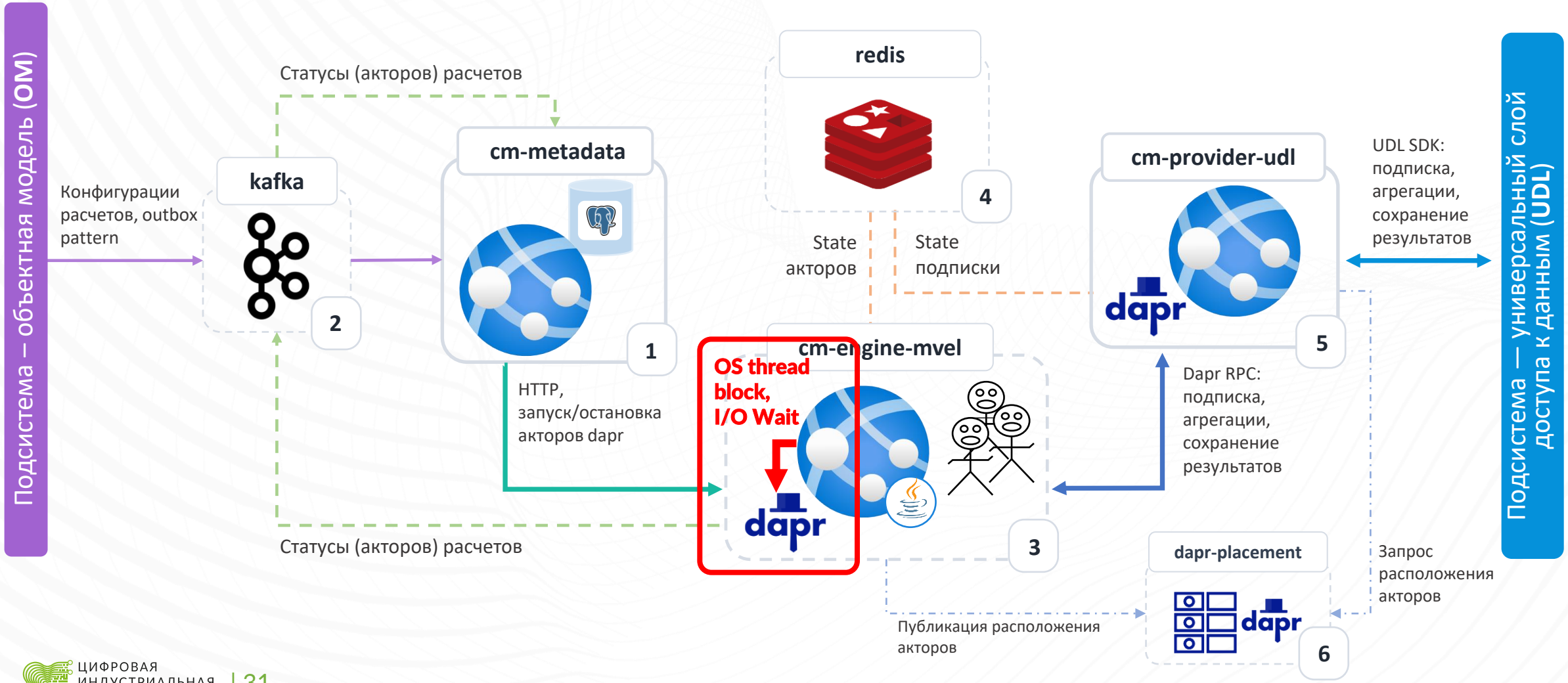
Выявленные проблемы первоначальной архитектуры

Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС

Подсистема расчётов (CM)

1. **cm-metadata** – конфигурация, управление, статусы расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, статусы расчётов (акторов)
3. **cm-engine-mvel** – исполнение MVEL- расчётов, хостинг акторов расчётов (java)
4. **redis** – Dapr state store. Акторы расчётов хранят в нём свое состояние
5. **cm-provider-udl** – сервис-фасад для UDL. Тоже использует Dapr state-store
6. **dapr-placement** – “DNS” для акторов Dapr



2023-09-11T12:35:49.054Z INFO MVEL eval result=9(Integer) for expression='A+B', context=
[Fn=all(MvelContextFunctions), A=4(Integer),
B=5(Integer)]","SourceContext":"com.zyfra.cm.services.eval.mvel2.MvelTaskExecutorImpl","Level":"Debug","A
ppName":"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml
2023-09-11T12:35:50.054Z INFO ExecutableStatement compilation completed for expression='A *
20'","SourceContext":"com.zyfra.cm.services.eval.mvel2.MvelExpressionCompilerImpl","Level":"Debug","AppNa
me":"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml
2023-09-11T12:35:50.054Z INFO Served stateless eval request {expression: \"A+B\" parameters { key: \"A\"
value { property_id: \"48c7d762-d3ae-11ed-9544-299d9fccf145\" value { string: \"4\" } } } parameters {
key: \"B\" value { property_id: \"48c7d761-d3ae-11ed-9544-299d9fccf145\" value { string: \"5\" } } }
timestamp { seconds: 1694435748 } result_type: VALUE_TYPE_DOUBLE} with result {result { double: 9.0
}}","SourceContext":"com.zyfra.cm.services.grpc.StatelessEvaluationServiceImpl","Level":"Debug","AppName"
:"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml"
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!

Unable to create more than 1024 Threads in OpenShift 4

SOLUTION VERIFIED - Updated May 23 2023 at 2:37 PM - English

Environment

- Red Hat OpenShift Container Platform (RHOCP)
  - 4

Issue

- How to change the value of `pids_limit` in OpenShift 4?
- Getting the following exception when migrating from OCP3 to OCP4:

Raw

```
java.lang.OutOfMemoryError: unable to create new native thread
```

Blocking calls for gRPC and HTTP even though using Reactor #43

Closed xiazuojie opened this issue on Jan 4, 2021 · 5 comments · Fixed by #445

xiazuojie commented on Jan 4, 2021 • edited by artursouza Contributor

Expected Behavior

Implementations of DaprClient API being non-blocking.

Actual Behavior

DaprClient API is reactive with Reactor. However, its implementations (gPRC and HTTP/1.x) at the moment are blocking, which defeats the purpose of using Reactive API.

US

Follow

artursouza Artur Souza

Maintainer **@dapr** | Engineering **@diagridio**

Washington State, USA

00:36 - 10h behind

Committed to this repository in the past month

Member of Dapr, and 2 more

were blocking for gRPC and HTTP.

artursouza commented on Jan 5, 2021 • edited

Member

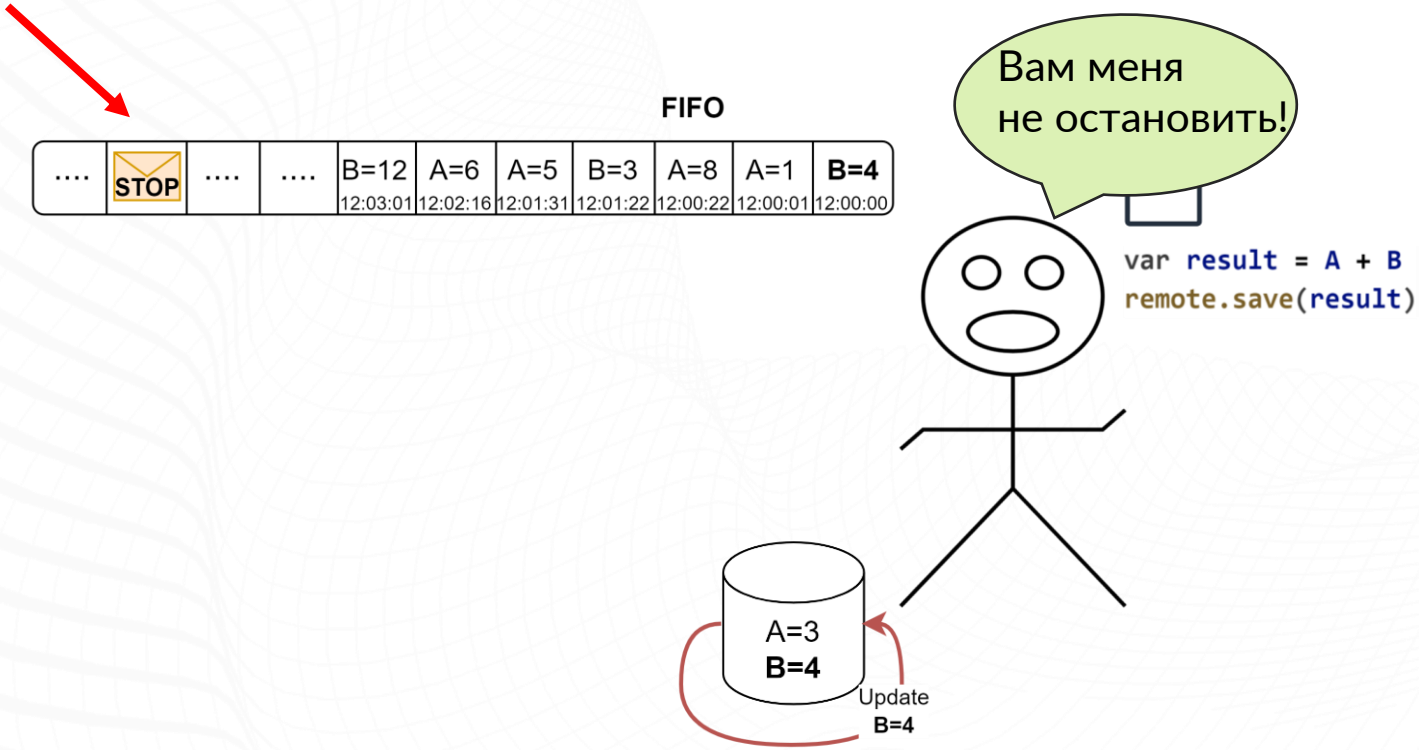
@xiazuojie Do you have a proposal to fix this? Since Java does not have async/await like .Net, I wonder how the Java community solves this.

Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode

Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode
3. Нагруженные расчёты “не остановить”. Команда на остановку становится в общую очередь сообщений актора расчёта

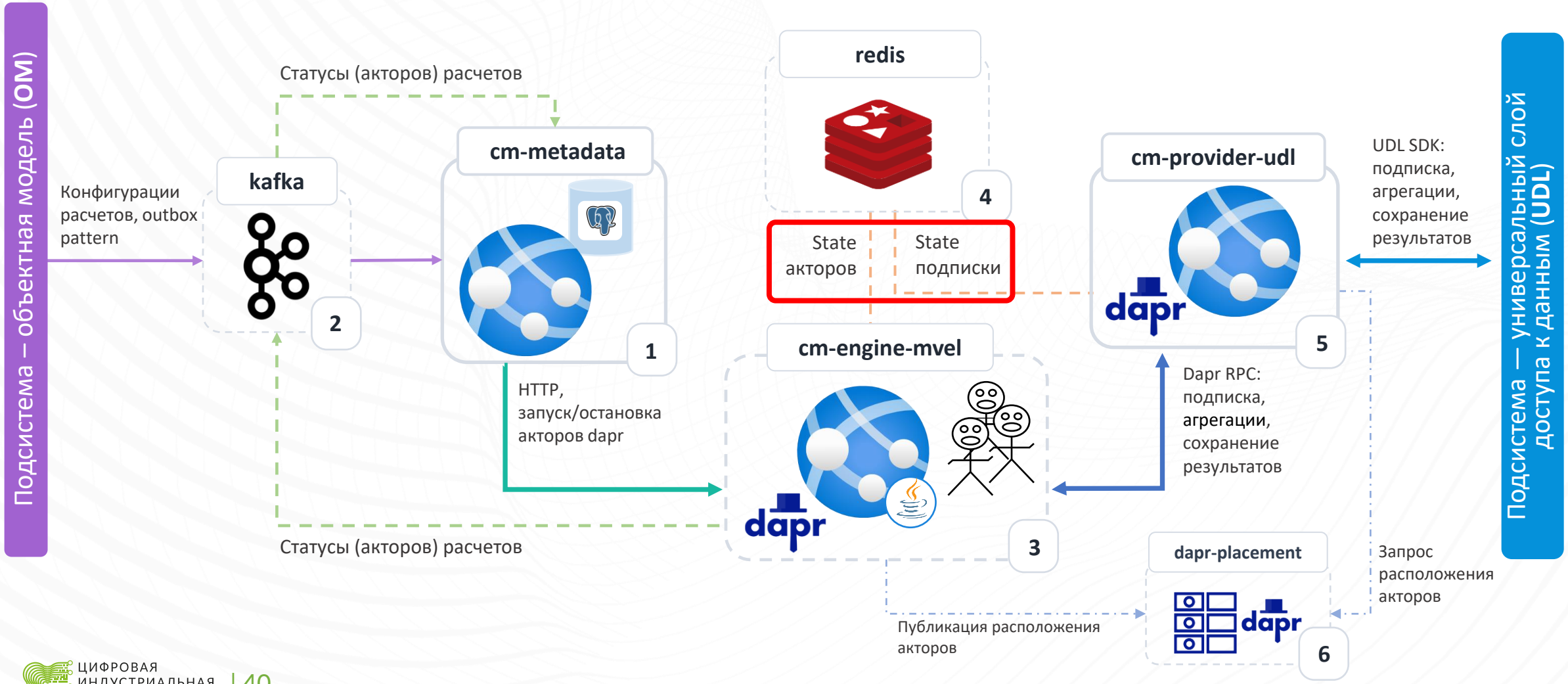

FIFO
....
STOP
....
....
B=12
12:03:01
A=6
12:02:16
A=5
12:01:31
B=3
12:01:22
A=8
12:00:22
A=1
12:00:01
B=4
12:00:00
Вам меня
не остановить!
var result = A + B
remote.save(result)
A=3
B=4
Update
B=4

Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode
3. Нагруженные расчёты “не остановить”. Команда на остановку становится в общую очередь сообщений актора расчёта
4. Нет транзакционности при инициализации актора расчёта. В краевых сценариях получаем живые, но неработающие акторы

Подсистема расчётов (СМ)

1. **cm-metadata** – конфигурация, управление, статусы расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из ОМ, статусы расчётов (акторов)
3. **cm-engine-mvel** – исполнение MVEL- расчётов, хостинг акторов расчётов (java)
4. **redis** – Dapr state store. Акторы расчётов хранят в нём свое состояние
5. **cm-provider-udl** – сервис-фасад для UDL. Тоже использует Dapr state-store
6. **dapr-placement** – “DNS” для акторов Dapr

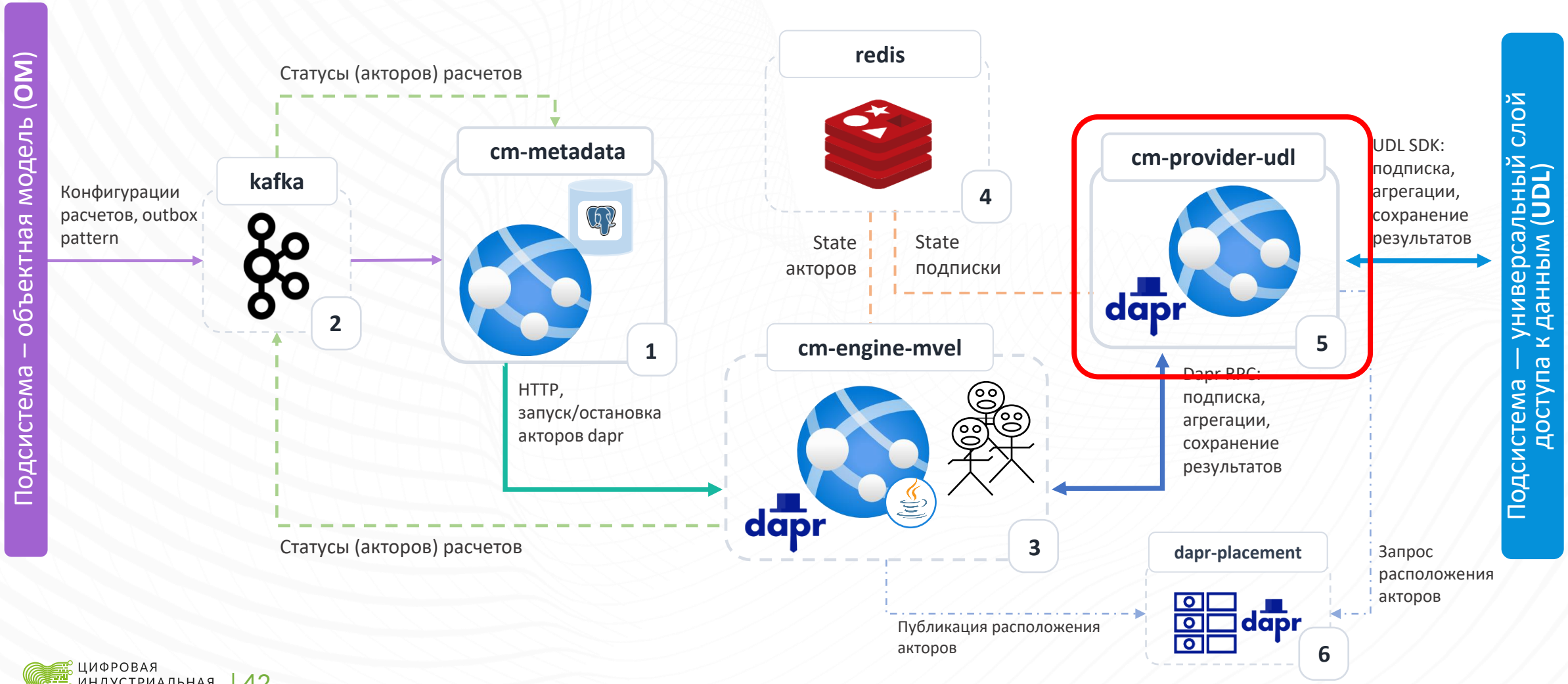


Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode
3. Нагруженные расчёты “не остановить”. Команда на остановку становится в общую очередь сообщений актора расчёта
4. Нет транзакционности при инициализации актора расчёта. В краевых сценариях получаем живые, но неработающие акторы
5. Компонент доступа к данным не поддерживает горизонтальное масштабирование

Подсистема расчётов (CM)

1. **cm-metadata** – конфигурация, управление, статусы расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, статусы расчётов (акторов)
3. **cm-engine-mvel** – исполнение MVEL- расчётов, хостинг акторов расчётов (java)
4. **redis** – Dapr state store. Акторы расчётов хранят в нём свое состояние
5. **cm-provider-udl** – сервис-фасад для UDL. Тоже использует Dapr state-store
6. **dapr-placement** – “DNS” для акторов Dapr

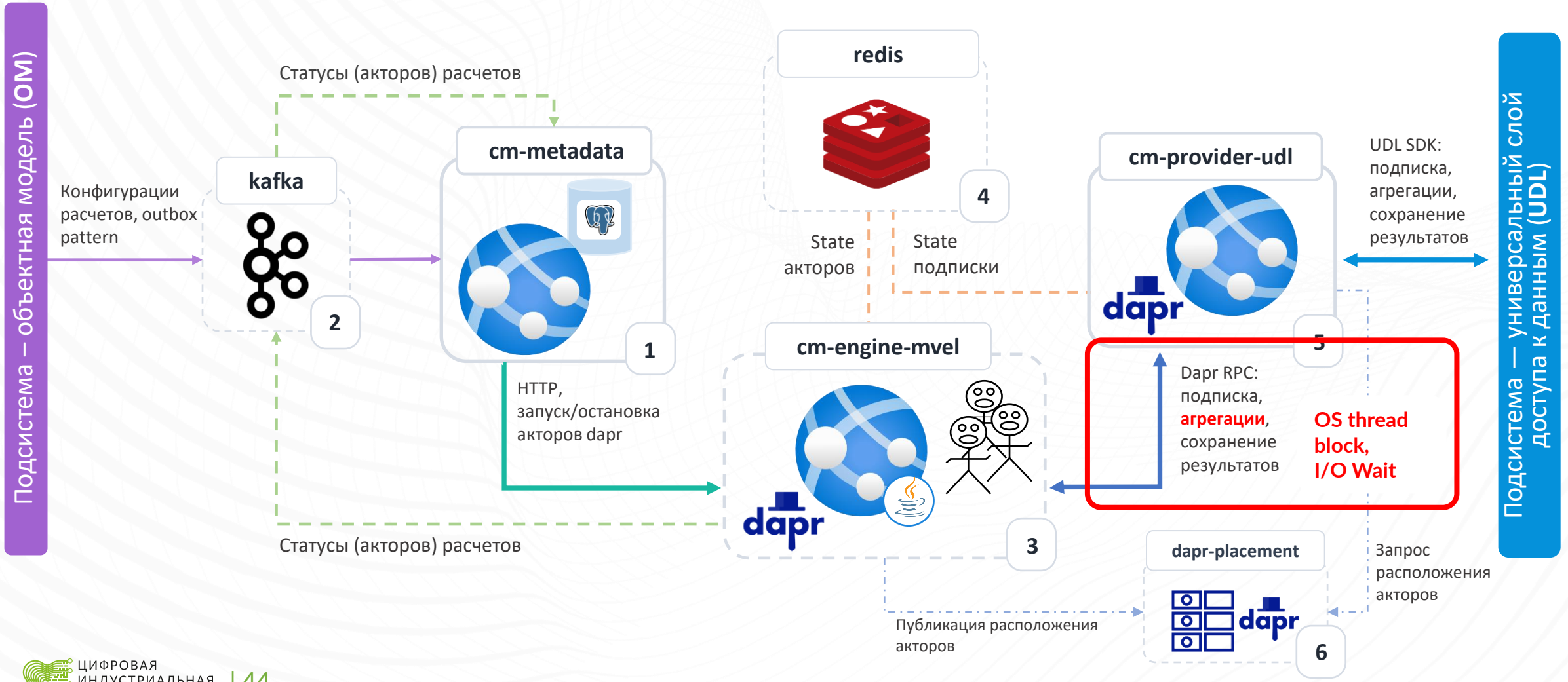


Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode
3. Нагруженные расчёты “не остановить”. Команда на остановку становится в общую очередь сообщений актора расчёта
4. Нет транзакционности при инициализации актора расчёта. В краевых сценариях получаем живые, но неработающие акторы
5. Компонент доступа к данным не поддерживает горизонтальное масштабирование
6. MVEL не умеет в реактивные подходы и неблокирующий I/O. Опять блокируются потоки ОС!

Подсистема расчётов (CM)

1. **cm-metadata** – конфигурация, управление, статусы расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, статусы расчётов (акторов)
3. **cm-engine-mvel** – исполнение MVEL- расчётов, хостинг акторов расчётов (java)
4. **redis** – Dapr state store. Акторы расчётов хранят в нём свое состояние
5. **cm-provider-udl** – сервис-фасад для UDL. Тоже использует Dapr state-store
6. **dapr-placement** – “DNS” для акторов Dapr



2023-09-11T12:35:49.054Z INFO MVEL eval result=9(Integer) for expression='A+B', context=
[Fn=all(MvelContextFunctions), A=4(Integer),
B=5(Integer)]","SourceContext":"com.zyfra.cm.services.eval.mvel2.MvelTaskExecutorImpl","Level":"Debug","A
ppName":"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml
2023-09-11T12:35:50.054Z INFO ExecutableStatement compilation completed for expression='A *
20'","SourceContext":"com.zyfra.cm.services.eval.mvel2.MvelExpressionCompilerImpl","Level":"Debug","AppNa
me":"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml
2023-09-11T12:35:50.054Z INFO Served stateless eval request {expression: \"A+B\" parameters { key: \"A\"
value { property_id: \"48c7d762-d3ae-11ed-9544-299d9fccf145\" value { string: \"4\" } } } parameters {
key: \"B\" value { property_id: \"48c7d761-d3ae-11ed-9544-299d9fccf145\" value { string: \"5\" } } }
timestamp { seconds: 1694435748 } result_type: VALUE_TYPE_DOUBLE} with result {result { double: 9.0
}}","SourceContext":"com.zyfra.cm.services.grpc.StatelessEvaluationServiceImpl","Level":"Debug","AppName"
:"zif-cm-engine-mvel.jar --spring.config.location=classpath:/application.yml"
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!
java.lang.OutOfMemoryError : unable to create new native Thread!

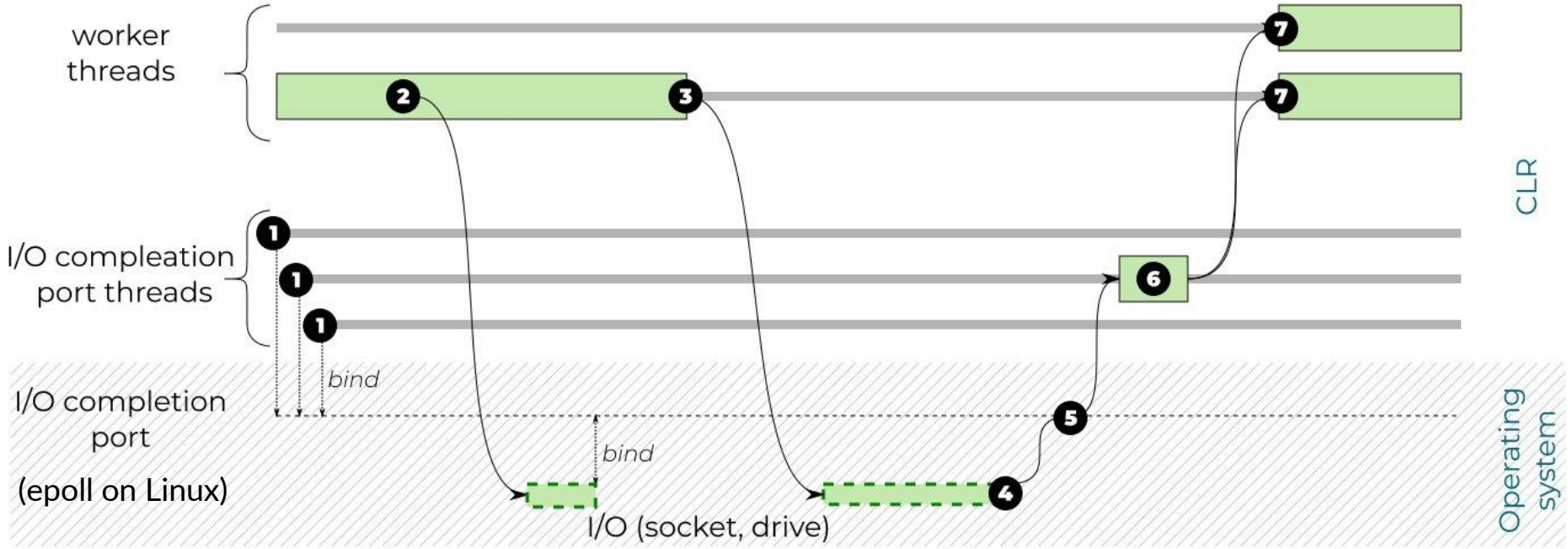
Отступление: Что, вообще, за тема с потоками ОС?

Потоки – дорогой ресурс ОС

- Занимает память под стек (до 8MiB на Linux);
- Создает нагрузку на планировщик ОС;
- Переключение между потоками требует переключения в контексте ядра ОС;
- Имеет PID, общее количество PID на сервере ограничено. (/proc/sys/kernel/pid_max)

> **Когда поток ожидает I/O, мы тратим этот дорогой ресурс**

Отступление: Что, вообще, за тема с потоками ОС?

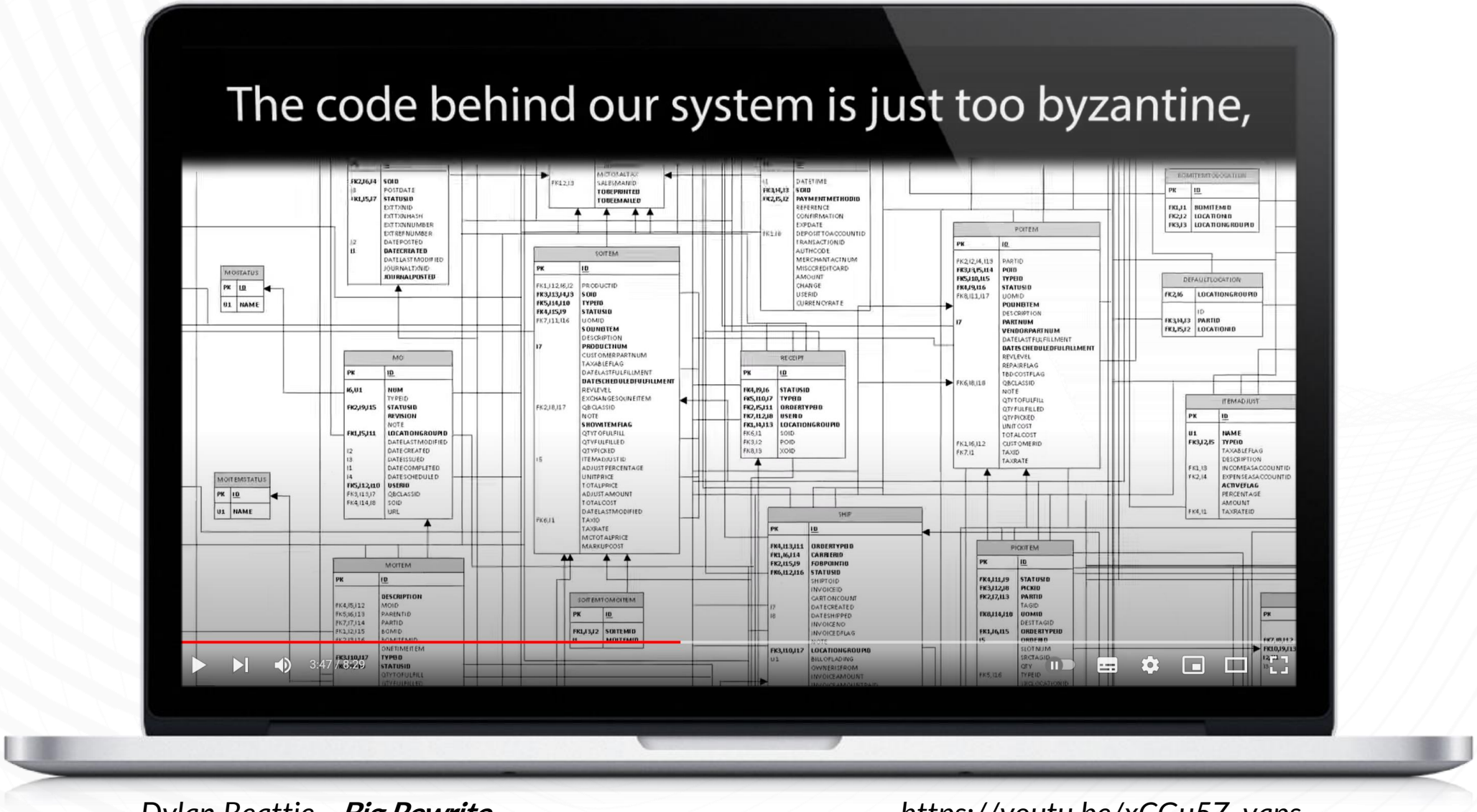


Подробнее – https://tooslowexception.com/net-asyncawait-in-a-single-picture

Выявленные проблемы первоначальной архитектуры

1. Сырая реализация Dapr SDK для Java. Блокируются потоки ОС
2. Споткнулись об “cloud-native”. Нет полномочий на деплой k8s-operator, деплоим в standalone mode
3. Нагруженные расчёты “не остановить”. Команда на остановку становится в общую очередь сообщений актора расчёта
4. Нет транзакционности при инициализации актора расчёта. В краевых сценариях получаем живые, но неработающие акторы
5. Компонент доступа к данным не поддерживает горизонтальное масштабирование
6. MVEL не умеет в реактивные подходы и неблокирующий I/O. Опять блокируются потоки ОС!

Нам нужно “Всё переписать” !



*Dylan Beattie - **Big Rewrite***

https://youtu.be/xCGu5Z_vaps

Проектирование новой архитектуры

Снова акторные фреймворки?

Orleans

proto.actor

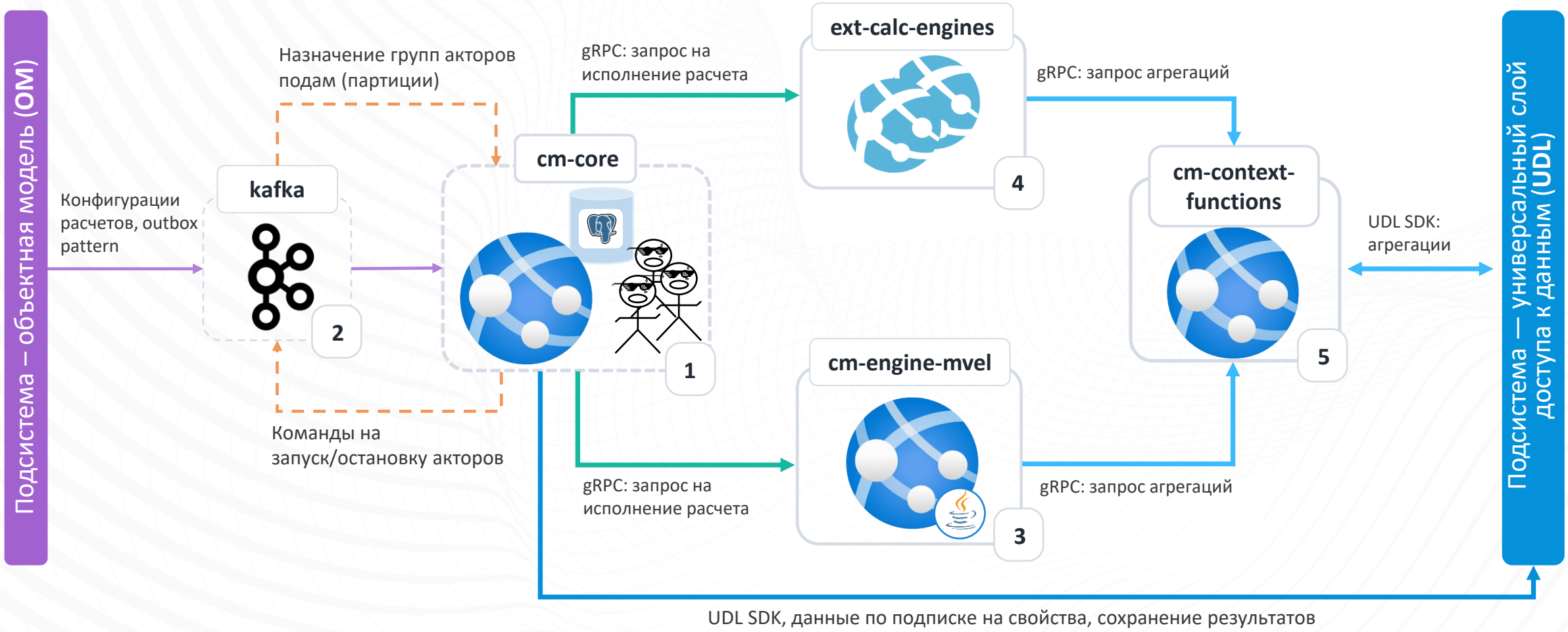
Свой узкоспециализированный акторный велосипед под задачу!

Требования:

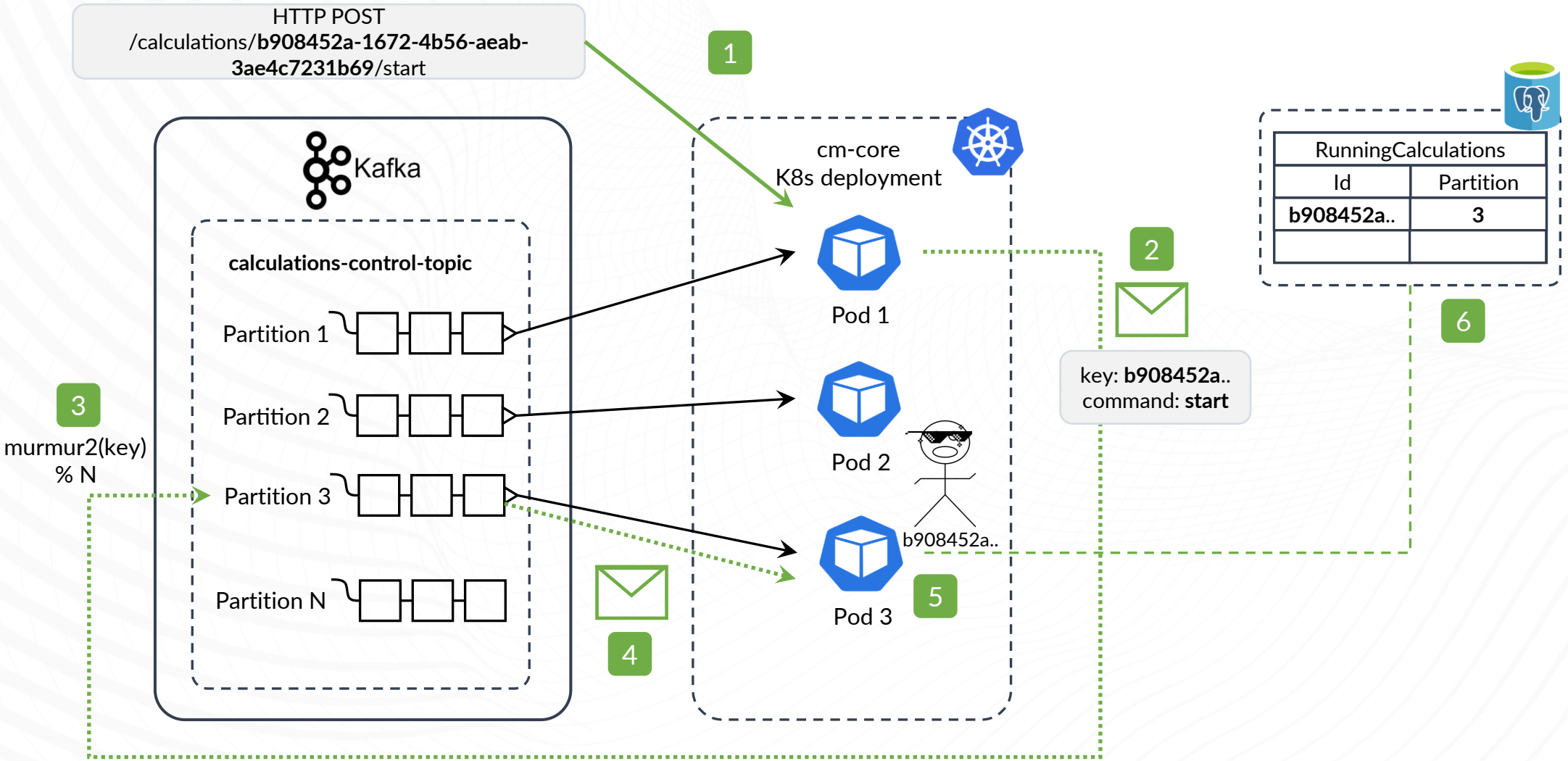
- Горизонтальное масштабирование, авторебаланс
- Восстановление состояния при перезапуске актора
- Поддержка новых функциональных требований (внешние сервисы, запуск по расписанию, по запросу)
- В идеале – использовать только уже имеющиеся инфраструктурные компоненты
- Обязательно – неблокирующий I/O

Подсистема расчётов (CM)

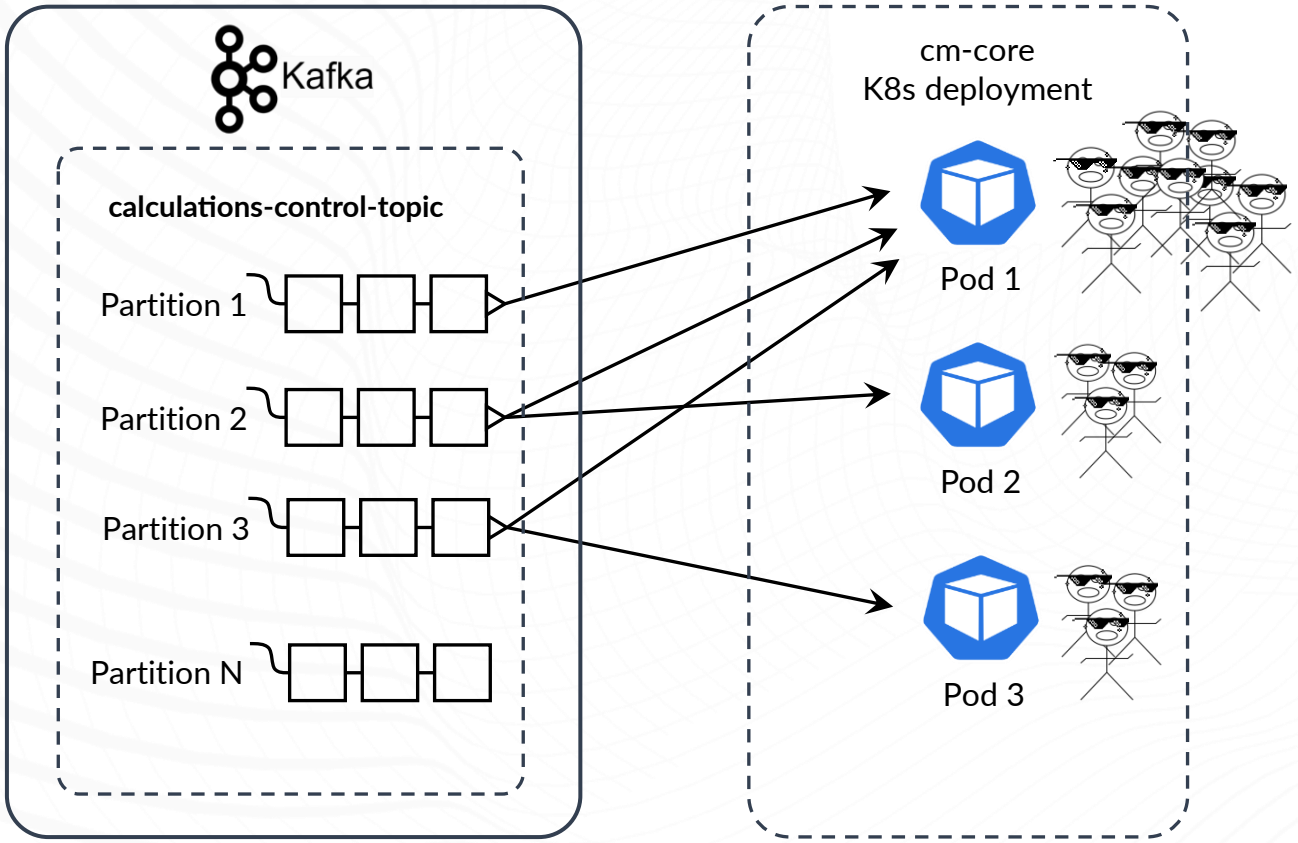
1. **cm-core** – (бывш. cm-metadata) – конфигурация, управление, статусы, хостинг акторов расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, назначение групп акторов, команды акторам
3. **cm-engine-mvel** – исполнение MVEL- расчётов (java)
4. **ext-calc-engines** – подключаемые внешние движки исполнения расчётов
5. **cm-context-functions** – выполнение агрегаций и других контекстных функций



Команды на запуск/остановку акторов, распределение по группам



Горизонтальное масштабирование, авторебаланс



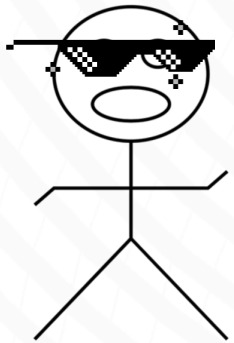
RunningCalculations	
Id	Partition
b908452a..	3
749ad54c..	2
B3ef6891..	1
Fa45e3ac..	3
NNNNN..	N

Горизонтальное масштабирование, авторебаланс

```
var consumerConfig = new ConsumerConfig
{
    BootstrapServers = KafkaConfiguration.KafkaBootstrapServers,
    GroupId = KafkaGroupId,
    ClientId = $"{_serviceName}-{Environment.MachineName}",
    AutoOffsetReset = AutoOffsetReset.Earliest,
    EnableAutoCommit = true,
    EnableAutoOffsetStore = false,
    PartitionAssignmentStrategy = PartitionAssignmentStrategy.CooperativeSticky
};
var consumer = ConsumerBuilder<Null, UpdateTaskStateCommand>(consumerConfig)
    .SetValueDeserializer(new KafkaJsonSerializer<UpdateTaskStateCommand>())
    .SetConsumerLogHandlers(_logger)
    .SetPartitionsAssignedHandler(PartitionsAssignedHandler)
    .SetPartitionsRevokedHandler(PartitionsRevokedHandler)
    .Build();
```

Восстановление состояния при перезапуске актора

Отдельный внешний StateStore не нужен, UDL отдаёт всё, что нужно!



Я запустился и хочу подписаться на **A и B**

Ты подписан. Стартовые значения **A=3, B=5**

Подсистема — универсальный слой доступа к данным (**UDL**)

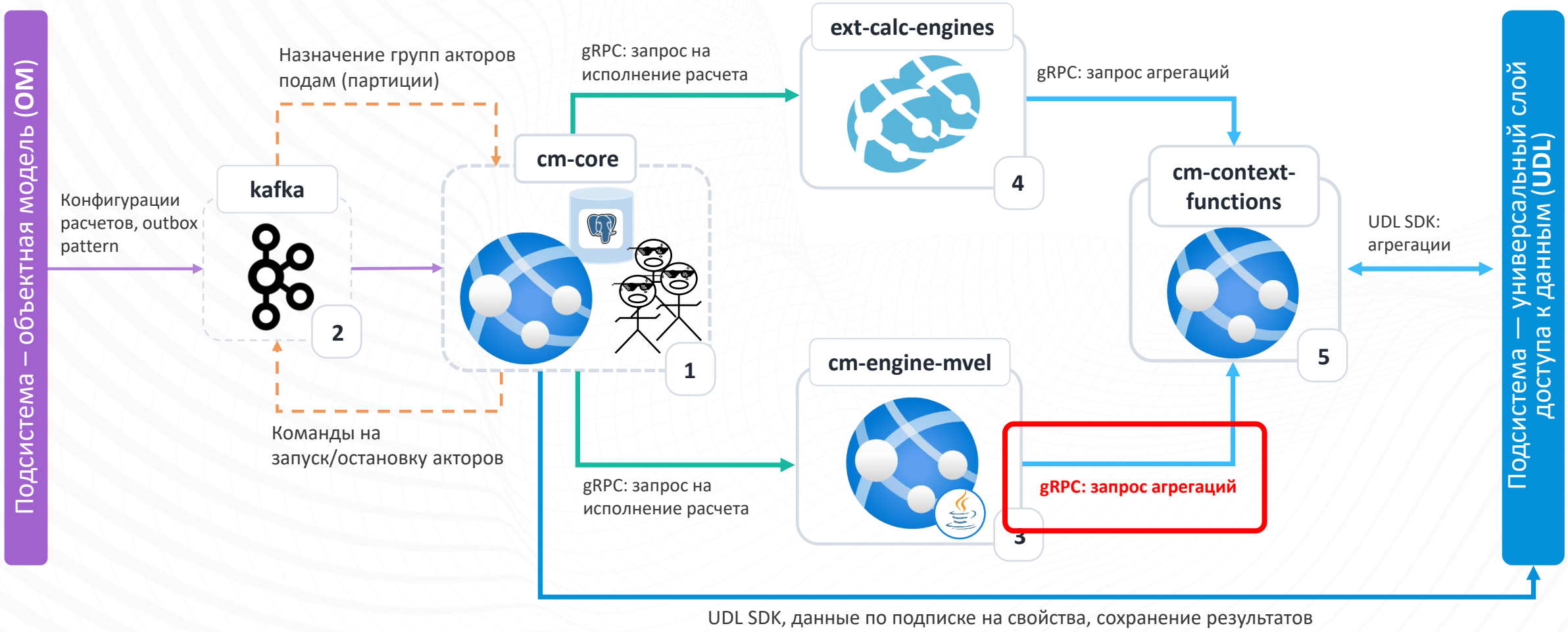
Свой узкоспециализированный акторный велосипед под задачу!

Требования:

- [x] Горизонтальное масштабирование, авторебаланс
- [x] Восстановление состояния при перезапуске актора
- [x] Поддержка новых функциональных требований (внешние сервисы, запуск по расписанию, по запросу)
- [x] В идеале – использовать только уже имеющиеся инфраструктурные компоненты
- [ ] Обязательно – неблокирующий I/O

Подсистема расчётов (CM)

1. **cm-core** – (бывш. cm-metadata) – конфигурация, управление, статусы, хостинг акторов расчётов
2. **kafka** – брокер сообщений. Конфигурации расчётов из OM, назначение групп акторов, команды акторам
3. **cm-engine-mvel** – исполнение MVEL- расчётов (java)
4. **ext-calc-engines** – подключаемые внешние движки исполнения расчётов
5. **cm-context-functions** – выполнение агрегаций и других контекстных функций



Project Loom спешит на помощь!

- Project Loom —
- Блокируем «зе
- Preview начина

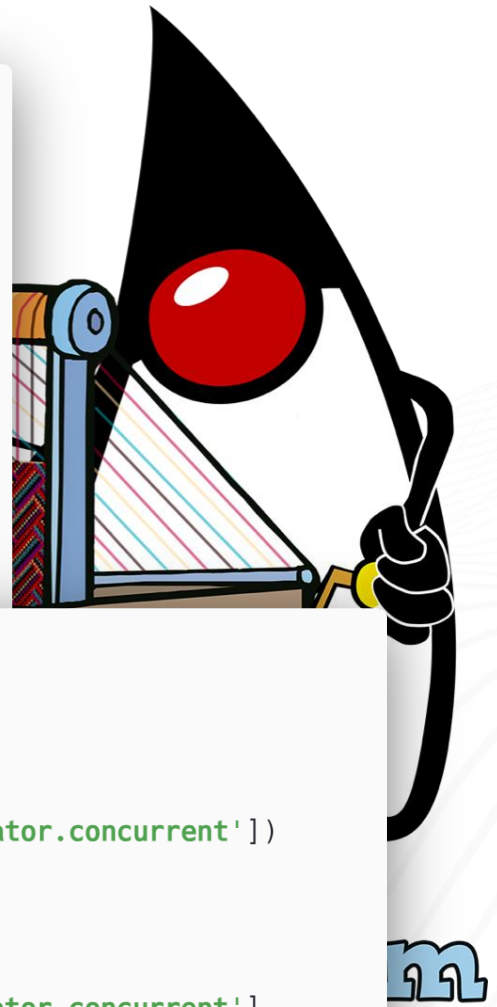
```java
import io.grpc.ServerBuilder;
import org.lognet.springboot.grpc.GRpcServerBuilderConfigurer;
import org.springframework.stereotype.Component;
import java.util.concurrent.Executors;

@Component
public class MainGRpcServerBuilderConfigurer extends GRpcServerBuilderConfigurer {
    @Override
    public void configure(ServerBuilder<?> serverBuilder){
        serverBuilder
                .executor(Executors.newVirtualThreadPerTaskExecutor())
                .build();

    }
}
```

```
compileJava {
    options.compilerArgs.addAll(['--release', '19'])
    options.compilerArgs.addAll(['--enable-preview'])
    options.compilerArgs.addAll(['--add-modules', 'jdk.incubator.concurrent'])
}

application {
    applicationDefaultJvmArgs = ['--enable-preview',
                                 '--add-modules', 'jdk.incubator.concurrent']
}
```



Результаты и выводы

Достигнутые результаты по производительности

Изначальная архитектура

Потребление на 70к расчётов по профилю НПЗ

Приложение	Количество экземпляров	Потребление ЦПУ (Ядер)	Потребление ОЗУ (ГиБ)
cm-metadata	10	1.2	1.9
cm-engine-mvel	32	2.1	5.8
cm-provider-udl	1	6	1.6
Итого:		85.2	206.2
Итого на 10к:		**12.2**	**29.4**

Новая архитектура

Потребление на 300к расчётов по профилю НПЗ

Приложение	Количество экземпляров	Потребление ЦПУ (Ядер)	Потребление ОЗУ (ГиБ)
cm-core	10	1.9	2.3
cm-engine-mvel	10	1	5.6
cm-context-functions	10	0.7	1.2
Итого:		36	91
Итого на 10к:		**1.2**	**3**

* Смежные подсистемы эмулировались и не подвергались нагрузке

** Intel Xeon Gold 6338, DDR4-3200

Выводы

1. Для своей задачи ложка удобнее, чем швейцарский нож. Часто лучший фреймворк – несуществующий

2. С осторожностью относитесь к новым модным технологиям с крутым лендингом, даже если они от вендоров-гигантов

Спасибо за внимание!