**МЕТОДИКА**

**ИЗНАЧАЛЬНЫЕ УСЛОВИЯ**

Изначальные технические(физические) условия: Существует сервер, имеющий память ??? ГБ и имеющий процессор марки - ???.

Планируется создание *микросервисного приложения*, состоящего из X = const сервисов, у каждого сервиса (пода) будет Ni, где i = [1, K] количество реплик (копий) сервиса, где K – заранее оговоренное некоторое число. Каждой реплике (копии) отводится некоторое пространство памяти. Ключевой особенностью сбора метрик заключается в анализе относительных, а не абсолютных значений. Введём новые термины:

*Нагрузочными тестами* будем называть упорядоченную конечную последовательность тестов L1 < …< Ln продолжительностью tсекунд, где n и t – заранее оговоренные числа.

*Статичными данными* будем называть данные, собранные экспериментальным путём с пары «система-сервер».

*Практическими данными* будем называть данные, полученные экспериментальным путём с пары «система-сервер» под действием нагрузочных тестов.

В предварительном этапе проводится измерение статичных данных:

* Статичная матрица задержки сети Mстат.сеть - симметричная матрица размерности N\*N, элементы отображают задержку сети между сервисами, по диагонали ставятся 0 или 1, так как задержка внутри отдельного сервиса минимальна.
* Относительная статичная загрузка памяти MEMстат (вектор значений размерности N, с элементами от 0 до 1);
* Относительная статичная загрузка процессора CPUстат (вектор значений размерности N, с элементами от 0 до 1);
* Абсолютное статичное энергопотребление системы Eстат (Квт\*ч).

Для системы в целом, и для отдельных сервисов (именно сервисов(подов), а не реплик), проводятся нагрузочные тесты, в результате которых логируются следующие величины:

* Нагрузочные тесты Li , где i = 1…n (вектор значений размерности N, единица измерения – запросы в секунду);
* Относительная загрузка памяти MEMi (вектор значений размерности N, с элементами от 0 до 1);
* Относительная загрузка процессора CPUi(вектор значений размерности N, с элементами от 0 до 1);
* Абсолютное энергопотребление системы Ei (Квт\*ч) //спорная величина;
* Матрица задержки сети Mсеть.i (симметричная матрица размерности N\*N, элементы отображают задержку сети между сервисами) P.S. по логике вещей должна быть такой же, как и статичная матрица задержки.
* Матрица времени отклика между отправкой запроса и получением ответа – Mload.i(матрица размерности N\*N, элементы которой отображают время между отправкой запроса и получением ответа).

Эксперимент проводится K раз, причем с каждым новым экспериментом количество реплик (копий сервисов) увеличивается.

Основными задачами данного эксперимента является ***выявления неявных закономерностей и определение оптимальной архитектуры*** микросервисного приложения.

Основными исследуемыми характеристиками являются:

1. Потребление памяти.
2. Потребление ресурса процессора.
3. Энергопотребление.
4. Стабильность сети.
5. Производительность.

**ПОТРЕБЛЕНИЕ ПАМЯТИ**

Для заданной нагрузки Li каждые t/r мс считывается вектор относительной загрузки памяти MEMi.r. Получим векторы

dMEMi.r = MEMi.r - MEMi.

Таким образом можно составить матрицу размерности N\*r, где строки матрицы отображают загруженность памяти при заданной нагрузке Li по времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, что можно будет говорить о средней загруженности памяти при нагрузке Li по времени t.

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

MEM = int(0, t)(f(x)dx)

Таким образом получим дискретную функцию

fmem(L) = MEM

Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

gmem(L) = MEM

а также получим функцию

g’mem(L) = dL/dg

**Оценка:**

* функция gmem(L) показывает зависимость памяти от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’mem(L) показывает рост загруженности памяти от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gmem.k(L), …, gmem.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост загруженности памяти (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gmem.k(L)dL)

Данный интеграл показывает нагрузочно-загрузочную характеристику памяти до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную нагрузочно-загрузочную характеристику памяти

Chi = Ii/i

Итоговая оценка заключается в Chi. **Чем она меньше, тем лучше.**

**ПОТРЕБЛЕНИЕ РЕСУРСА ПРОЦЕССОРА**

*Аналогична оценке потребления памяти.* Для заданной нагрузки Li каждые t/r мс считывается вектор относительной загрузки процессора CPUi.r. Получим векторы

dCPUi.r = CPUi.r - CPUi.

Таким образом можно составить матрицу размерности N\*r, где строки матрицы отображают загруженность процессора при заданной нагрузке Li по времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, что можно будет говорить о средней загруженности процессора при нагрузке Li по времени t.

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

CPU = int(0, t)(f(x)dx)

Таким образом получим дискретную функцию

fCPU(L) = CPU

Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

gCPU(L) = CPU

а также получим функцию

g’CPU(L) = dL/dg

**Оценка:**

* функция gCPU(L) показывает зависимость загруженности процессора от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’CPU(L) показывает рост загруженности процессора от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gCPU.k(L), …, gCPU.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост загруженности процессора (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gCPU.k(L)dL)

Данный интеграл показывает нагрузочно-загрузочную характеристику процессора до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную нагрузочно-загрузочную характеристику памяти

Chi = Ii/i

**ЭНЕРГОПОТРЕБЛЕНИЕ**

Для заданной нагрузки Li каждые t/r мс считывается вектор относительной загрузки процессора CPUi.r. Получим векторы

dPi.r = Pi.r - Pi.

**СТАБИЛЬНОСТЬ СЕТИ**

**ПРОИЗВОДИТЕЛЬНОСТЬ**

**ОБЩИЕ ВЗАИМОСВЯЗИ**

Потенциальные выводы, которые можно будет сделать на основе полученного датасета.

**Нагрузки:**

**----**

**Производительность:**

Нагрузка памяти: вычисляются векторы

dMEMi = MEMi - MEMстат

Получаемые значения образуют функцию:  
 fmem(L) = dMEMi

Функцию

Нагрузка процессора: вычисляются векторы

dCPUi = CPUi - CPUстат

Энергопотребление: вычисляется следующий скаляр

dEi = Ei - Eстат

**Стабильность:** Стабильность сети: вычисляется матрица разности статической матрицы задержки сети и матрицы задержки сети

dMсеть.i = Mсеть.i - Mстат.сеть

затем вычисляется дисперсия матрицы разности:

Dсеть.i = dis(dMсеть.i)

В итоге получается дискретная функция, отражающая зависимость дисперсии от нагрузочных тестов:  
 fсеть (L) = Dсеть

На основе дискретных данных аппроксимацию получим непрерывную, всюду дифференцируемую функцию gсеть(L) :

gсеть(L) ~ fсеть(L)

Затем получим функцию g’сеть = dL/dg

Исходя из вычисленных величин сделаем следующие выводы:

* Функция gсеть(L) носит оценочный характер, она позволяет понять общий показатель стабильности/нестабильности сети. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* Функция g’сеть показывает, начиная с какой нагрузки рост нестабильности сети становится критичным для всей системы. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*