SAMARA UNIVERSITY

Востокин С.В., Русин М.А.

Самарский национальный исследовательский университет
имени академика С.П. Королева

ЭКСПЕРИМЕНТЫ С ГЕНЕРАТОРОМ ПОСЛЕДОВАТЕЛЬНОСТИ A022008 ДЛЯ ИЗУЧЕНИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ СЕРВИСА СИНХРОНИЗАЦИИ СОСТОЯНИЯ

X Международная конференция «Суперкомпьютерные дни в России»
г. Москва, Шуваловский корпус МГУ, 22 – 23 сентября 2024 года

1. Мотивация исследования.
2. Архитектура сервиса синхронизации состояния.
3. Распределенный алгоритм генерации последовательности A022008.
4. Программная реализация и методика экспериментов.
5. Результаты экспериментов.
6. Выводы.

Задачи современных цифровых предприятий, связанные с обработкой операционных данных, обучением моделей AI для управления производством и бизнес-процессами, могут успешно решаться средствами грид-систем уровня предприятия.

Обычно у цифрового предприятия

- возникают ***ресурсоемкие вычислительные задачи***, связанные с его деятельностью;
- имеется ***большой парк вычислительной техники***, объединенный скоростной сетью.

Примеры: Самарский университет и другие вузы, ОДК Кузнецов, ЦСКБ Прогресс …

Агрегация вычислительных ресурсов в ресурсоемких приложениях традиционно реализуется на базе грид-платформ, основанных на ***использовании парадигмы задач***.

Примеры: BOINC, HTCondor, Dirac, Everest …

Преимущества парадигмы задач:

- ***универсальный подход***, пригодный для разнотипных приложений;
- универсальный способ обеспечить ***отказоустойчивость вычислений***;
- ***задачи можно объединить*** с использованием графов зависимостей, а грид-платформа может поддерживать отказоустойчивое выполнение таких графов;
- и др.

Однако построение грид-систем уровня предприятия на основе парадигмы задач имеет ***ряд значимых ограничений***:

- высокие ***накладные расходы на запуск задач***, что вынуждает увеличивать время выполнения задачи за счет увеличения объема вычислений для поддержания эффективности;
- длительные автономные задачи увеличивают ***накладные расходы на миграцию*** нагрузки, которая может выполняться непредсказуемо часто в сети предприятия;
- ***сложность механизма планирования*** задач влечет усложнение архитектуры платформы грид-вычислений в целом;
- грид-приложение является многокомпонентным, со ***связями через CLI-интерфейс***, что не вполне удобно при разработке и развертывании.

Пример: метод пилотных заданий и двухуровневое планирование задач - частичный способ преодоления перечисленных ограничений популярных грид-платформ.

В работе исследуется более радикальный подход организации вычислений в гриде уровня предприятия, основанный на ***отказе от использования парадигмы задач***, по крайней мере на уровне грид-платформы.

Вместо парадигмы задач предлагается ***использовать парадигму журнала событий*** и ***синхронизации глобального состояния*** компонентов распределенного приложения на его основе.

Обоснование применимости предлагаемой концепции грид-платформы предполагает:

- разработку ***проекта архитектуры*** грид-платформы;
- разработку методов создания ***грид-приложений*** для грид-платформы;
- ***исследование эффективности*** грид-платформы на типовой нагрузке и задаче.

Принцип вычислений: если известно начальное состояние вычислений и последовательность изменений состояния, то текущее состояние может быть вычислено независимо любыми наблюдателями.

Сервис предоставляет компонентам распределенного приложения:

- ***синхронизацию*** при записи изменений локальных состояний;
- ***отказоустойчивость при записи*** с семантикой доставки at-least-once;
- ***отказоустойчивое хранение*** записей, доставленных в журнал событий.

Дополнительно:

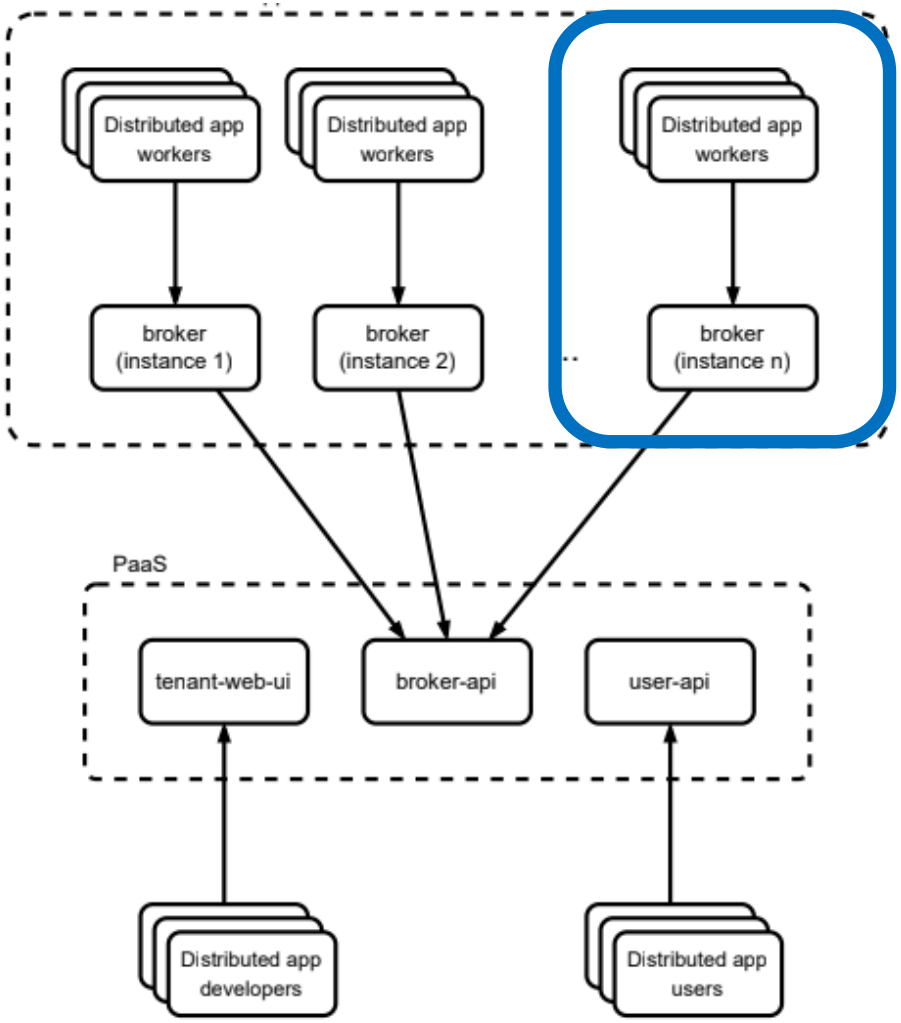
- предусматривается ***механизм аутентификации*** компонентов приложения и ***управления их правами доступа*** к журналу событий приложения;
- предусматривается ***механизм взаимодействия*** приложения ***с программным окружением*** - с процессами пользователя и между приложениями сервиса.

Важно: сервис “собирается” из готовых компонентов с проверенными механизмами отказоустойчивости - СУБД, встраиваемая БД, брокер сообщений, фреймворк RPC, веб-сервер и др.

Предлагается вариант трехуровневой архитектуры, с учетом возможностей

- масштабируемости,
- гибкости разработки,
- удобства тестирования,
- эффективность разработки.

На текущем этапе разработки сервиса исследовались возможности брокера, использующего кэш журнала событий на базе встраиваемой БД SQLite



Алгоритм исследует в восходящем порядке все простые числа, не превосходящие заданное, с целью обнаружения шестерок вида **(p-16, p-12, p-10, p-6, p-4, p)** из близко расположенных простых чисел.

Число **p-16** является очередным членом последовательности ***A022008 OEIS*** из Онлайн-энциклопедии целочисленных последовательностей Нила Слоуна.

Важные свойства алгоритма:

- простая и понятная последовательная реализация;
- моделирует семейство алгоритмов проекта PrimeGrid
  - .. и в целом семейство поисково-переборных алгоритмов;
- при увеличении диапазона проверки быстро растет сложность;
- вычисление очередного числа использует результаты вычислений предыдущих чисел,
  - .. что обосновывает применение синхронизации глобального состояния.

Алгоритм подпроцесса журнала событий

```
algorithm 'event log subprocess'
state variables:
 log := {}
 last_event := 0
-----------------------------------------------------------
while ⌉(stopped_by_the_service_operator()) do
|  receive message
|  if contains_the_results(message) then
|  | last_event := last_event + 1
|  | log[last_event] := extract_the_results(message)
|  end
|  answer := form_answer(log,message.events_from,last_event)
|  send answer
end while
-----------------------------------------------------------
```

Алгоритм вычислительного подпроцесса

```
algorithm 'calculation subprocess'
constant:
 N - number of search ranges
state variables:
 planned_tasks := {0,1,2,..,N-1}
 completed_tasks := {}
 P := 0 - last printed task
 odd_prime_table := {3}
 last_known_event := 0
-------------------------------------------------
 message.events_from := 1
 message.results := {}
 send message
 call sub algorithm 'update and print results'
```

```
while planned_tasks ≠ {} do
| id := get_random_element(planned_tasks)
| search_range := get_range(id)
| odd_prime_table :=
|          update(odd_prime_table,sqrt(search_range.upper))
| search_result := find(search_range,odd_prime_table)
| message.events_from := last_known_event + 1
| message.results := search_result
| send message
| call sub algorithm 'update and print results'
end while
-------------------------------------------------------------
```

Распечатка найденных членов последовательности

```
sub algorithm 'update and print results'
 external state variables:
  P, planned_tasks, completed_tasks, last_known_event
 ---------------------------------------------------------
 receive answer
 tasks_ids := get_ids(answer.completed_tasks)
 last_known_event := answer.last_event
 planned_tasks := planned_tasks \ tasks_ids
 completed_tasks := completed_tasks ∪ answer.completed_task
 T := P
 while completed_tasks ≠ {} ∧
has_task_with_id(completed_tasks,T+1)do
 | task := get_task_by_id(completed_tasks,T+1)
 | print_task_results(task)
 | completed_tasks := completed_tasks \ {task}
 | T := T+1
 end while
 P := T
 ---------------------------------------------------------
```

Исследуется влияние на производительность

- необходимости доставки записей об изменении локальных состояний каждому рабочему процессу;
- избыточности вычислений, вызванной случайным выбором диапазона чисел для поиска членов последовательности A022008.

Программные реализации:

- гибридный имитационный эксперимент – C++, Templet SDK
  - накладные расходы алгоритма при отсутствии сетевых задержек и заданных сетевых задержках;
- программа нагрузочного тестирования – C#, gRPC, SQLite
  - проверка отказоустойчивости, абсолютное ускорение.

Аппаратура грид-системы предприятия – Медиацентра Самарского университета:

- узел: Intel(R)Core(TM)i5-10400 @2.9 GHz (6 cores) L1 384 KB L2 1.5 MB L3 12 MB RAM 8.00 GB with Windows 10 Pro 22H2;
- сеть: Evernet 100 MB/s.

Сравнение абсолютного и относительного ускорения без учета задержек на сервере (гибридная имитация, 10 узлов)

	Кол-во интервалов				
Ускорение	100	200	300	400	500
относительное, T1rel/Tp	9,75	9,80	9,90	9,92	9,94
абсолютное, T1abs/Tp	7,12	8,32	9,22	9,34	9,38
перевычисления, T1rel/T1abs	1,37	1,18	1,07	1,06	1,06

Область поиска - до 100.000.000 (82 первых членов последовательности A022008)
T1abs = 148 с (на виртуальной машине в облаке Медиацентра Самарского университета)

Относительное ускорение с учетом задержек на сервере (гибридная имитация, 10 узлов)

	Задержка на сервере, с				
Кол-во интервалов	0,05	0,04	0,03	0,02	0,01
100	9,50	9,63	9,72	9,60	9,72
200	9,35	9,49	9,63	9,73	9,82
300	8,90	9,33	9,50	9,75	9,81
400	8,08	8,93	9,39	9,63	9,84
500	6,66	8,19	9,11	9,54	9,80

Абсолютное ускорение с учетом задержек на сервере (гибридная имитация, 10 узлов)

	Задержка на сервере, с				
Кол-во интервалов	0,05	0,04	0,03	0,02	0,01
100	7,09	7,23	7,26	7,42	7,13
200	7,20	7,52	7,66	8,01	8,47
300	7,07	7,48	7,92	8,39	8,61
400	6,22	6,92	7,73	8,37	8,64
500	5,30	6,47	7,76	8,02	8,75

Абсолютное ускорение при выполнении в гриде предприятия

	Кол-во интервалов				
Кол-во узлов	100	200	300	400	500
10	6,18	6,47	6,26	5,78	5,54
50	21,36	24,69	24,97	24,36	23,93

T1 = 213 с (82 первых членов последовательности A022008)
Tp_min = 8,5 с (при разбиении области до 100.000.000 на 300 интервалов)

1. Сервис позволяет гибко конфигурировать приложение, не прерывая вычисления, добавляя или удаляя его рабочие процессы.
2. Увеличение числа разбиений поискового пространства уменьшает долю избыточных вычислений и повышает ускорение, а приемлемый уровень избыточных вычислений достигается при 30 разбиениях на процесс и далее заметно не снижается (слайд 14).
3. Если не учитывать объем избыточных вычислений (слайд 15), то наблюдается высокое ускорение при всех исследованных сочетаниях задержек сервера и числа разбиений.
4. Причиной уменьшения ускорения (слайд 15, 16) является доля последовательных вычислений, вызванная сетевой задержкой, что объясняется законом Амдала.
5. В реальных окружениях с сетевой задержкой (слайд 17), если увеличение числа разбиений увеличивает ускорение, следует продолжать увеличение, пока не будет достигнуто максимальное значение ускорения.

Эксперименты подтверждают применимость предложенной архитектуры сервиса синхронизации состояния на основе журнала событий для управления вычислениями в гриде предприятия.

http://templet.ssau.ru/wiki - вики и образовательные ресурсы проекта Templet

https://github.com/the-templet-project/templet/tree/master/samples/primesix – исследуемый код и результаты экспериментов

Авторы: *Востокин Сергей Владимирович* (easts@mail.ru)
д.т.н., зав. кафедрой программных систем, Самарский национальный исследовательский университет имени академика С.П. Королева

Русин Максим Алексеевич
аспирант кафедры программных систем, Самарский национальный исследовательский университет имени академика С.П. Королева

СПАСИБО ЗА ВНИМАНИЕ !