

Новосибирский государственный университет

09.03.01 Информатика и вычислительная техника. Программная инженерия и компьютерные науки

Разработка и реализация алгоритмов повышения доступности данных в системе LuNA

Выполнил: Масыч М. А.

Руководитель ВКР: д.т.н., профессор зав. каф. ПВ ФИТ НГУ Малышкин В. Э.

Соруководитель ВКР: ст. преп. каф. ПВ ФИТ Перепёлкин В. А.

Рецензент: к.ф.-м.н., с.н.с. ИМ СО РАН Горкунов Е. В.

23.06.2021

Актуальность

Важной и трудоемкой частью разработки параллельных программ является управление данными. В распределенных вычислительных системах обеспечение доступности данных, поддержка сбалансированного распределения данных по процессорным элементам и приемлемой скорости доступа к ним — нетривиальные задачи.

В общем случае, для задачи управления распределенными данными не существует универсального алгоритма.

Одной из актуальных подзадач в области управления данными является проблема доступности данных.

Обзор

Разработка алгоритмов управления распределенными данными — активная сфера научной деятельности. Невозможность разработки универсального алгоритма является стимулом для постоянного возникновения новых разнообразных решений данной проблемы. Примерами алгоритмов управления распределенными данными могут служить:

- Диффузионные алгоритмы;
- ASURA;
- RUSH;
- NARA;
- Rope-of-Beads.

Многие алгоритмы управления распределенными данными стремятся к хорошим средним результатам в своей области применения. Однако, неизбежно теряют эффективность на некоторых конкретных задачах.

LuNA

Система фрагментированного программирования LuNA является удобной средой для развития и накопления подобных алгоритмов, являясь средством для решения разнообразных распределенных задач и предоставляя удобные средства разработки.

Цели и задачи

Целью работы является разработка и реализация алгоритмов повышения доступности данных в системе LuNA.

Для достижения цели поставлены следующие задачи:

1. Проведение анализа существующих решений;
2. Формулировка требований;
3. Разработка алгоритмов;
4. Реализация алгоритмов;
5. Тестирование.

Контекст

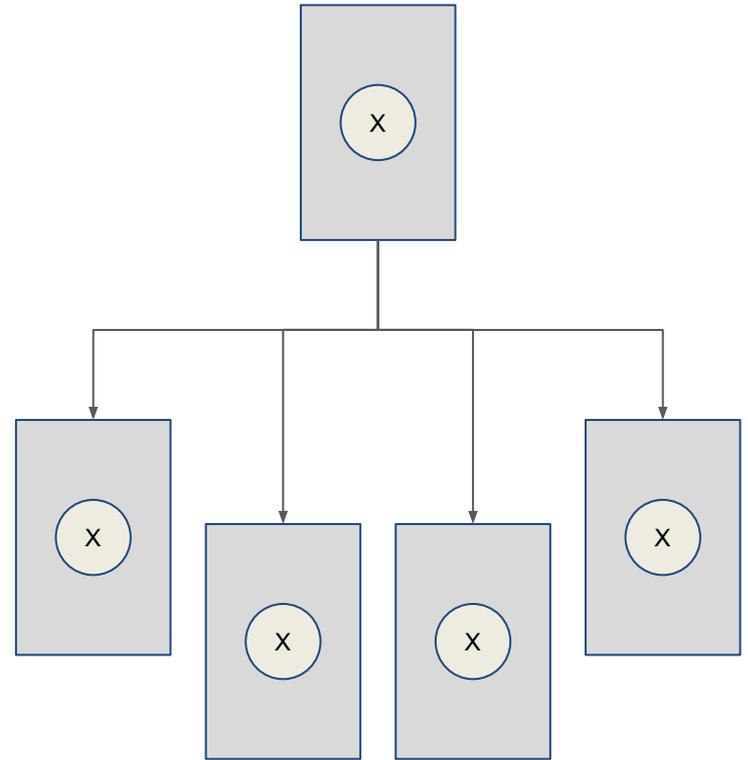
Программа на LuNA состоит из множества фрагментов вычислений. Фрагменты вычислений — это некоторый исполняемый код, который принимает на вход какое-либо количество фрагментов данных и порождает какие-либо фрагменты данных на выходе.

Фрагменты данных представляют собой некоторые переменные, структуры или массивы данных.

Фрагменты могут мигрировать между узлами вычислительной системы. На данный момент существует два механизма перемещения данных: данные хранятся на узле и предоставляются по запросу или данные отправляются по готовности потребителю.

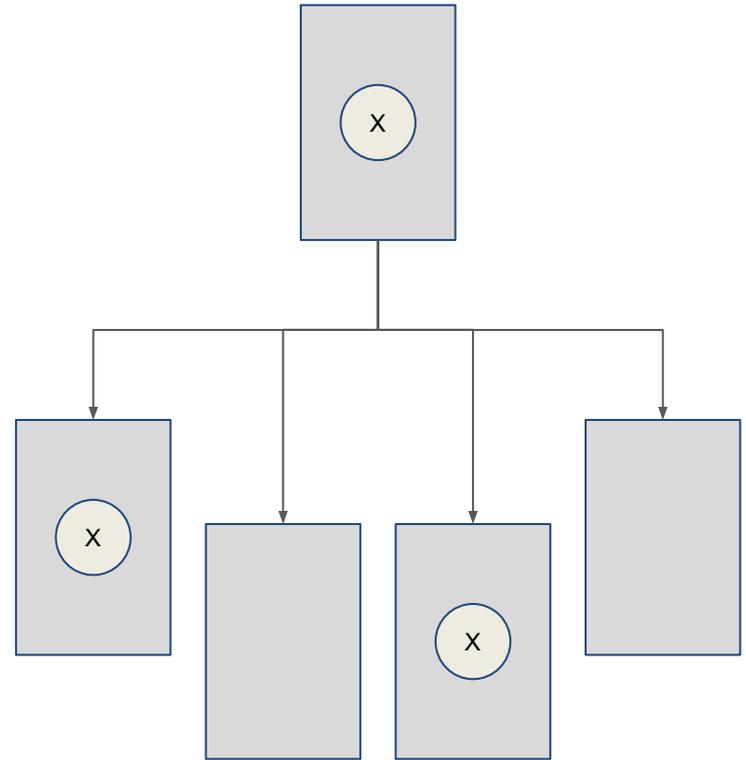
Массовая репликация

- Копирование фрагмента данных с одного узла на все остальные узлы системы.
- Подобное решение закрывает проблему поиска данных и обеспечения приемлемой скорости их получения — фрагменту вычислений не нужно отправлять запрос на получение отсутствующего фрагмента данных другим узлам (фрагмент уже есть или окажется на узле).



Ограниченная репликация

- Копирование фрагмента данных с одного узла на часть узлов системы.
- Снижение затрат на поиск и получение данных.
- Снижение нагрузки на память и сеть системы.



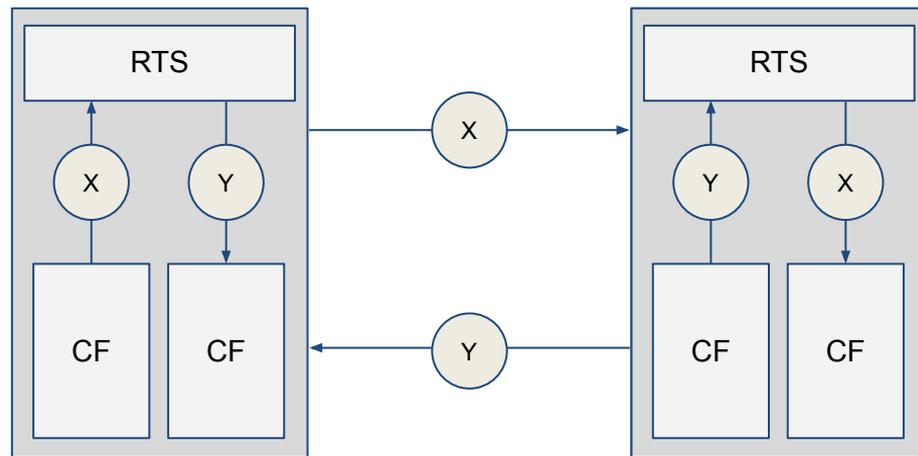
Кэширование

- Репликация данных оказывает высокую нагрузку на память системы.
- Целесообразно иметь возможность удалять данные как вручную, так и автоматически, по истечению какого-либо числа использований.

Реализация

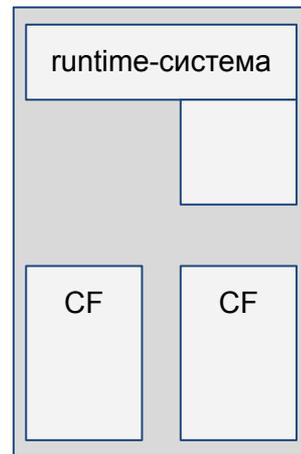
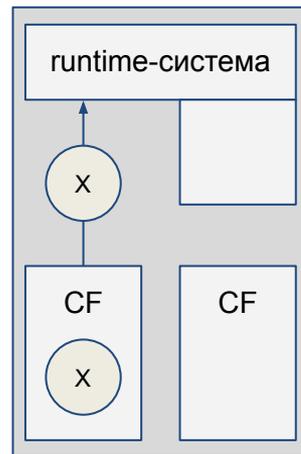
Runtime-система LuNA позволяет фрагментам вычислений отправлять и принимать фрагменты данных (в том числе между узлами) через класс `rts`.

Для каждого из предложенных алгоритмов, класс `rts` был расширен новыми методами. Кроме того, был также расширен класс `comm`, через который `rts` отправляет сообщения в распределенной системе.



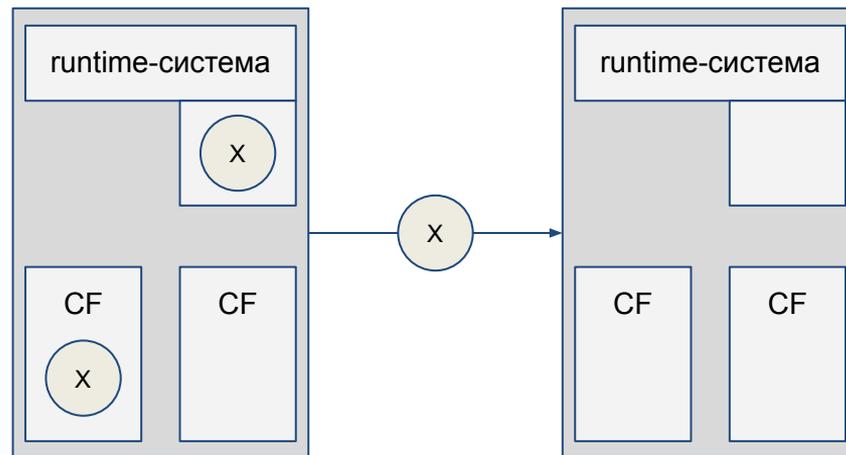
Массовая репликация

1. Фрагмент вычислений обращается к runtime-системе на своем узле.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его всем остальным узлам.
3. Фрагменты вычислений запрашивают у своих узлов нужный фрагмент данных.
4. Как только узел получит требующийся фрагмент данных, он передаст его ожидающим фрагментам вычислений.



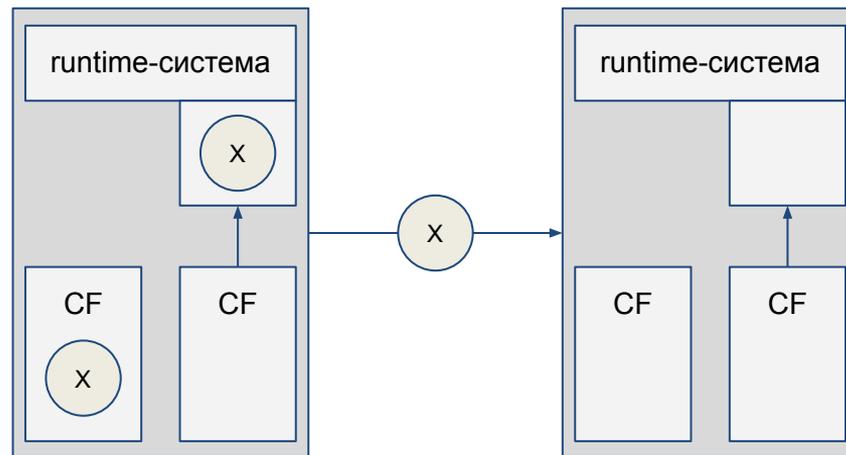
Массовая репликация

1. Фрагмент вычислений обращается к runtime-системе на своем узле.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его всем остальным узлам.
3. Фрагменты вычислений запрашивают у своих узлов нужный фрагмент данных.
4. Как только узел получит требующийся фрагмент данных, он передаст его ожидающим фрагментам вычислений.



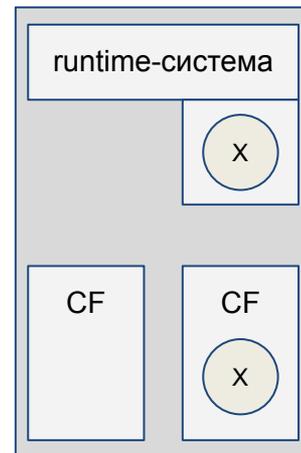
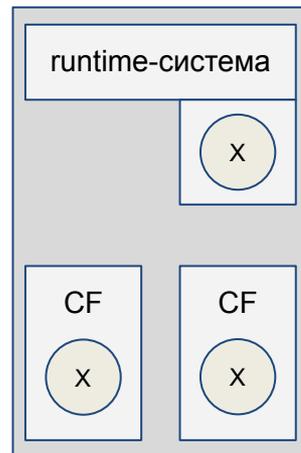
Массовая репликация

1. Фрагмент вычислений обращается к runtime-системе на своем узле.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его всем остальным узлам.
3. Фрагменты вычислений запрашивают у своих узлов нужный фрагмент данных.
4. Как только узел получит требующийся фрагмент данных, он передаст его ожидающим фрагментам вычислений.



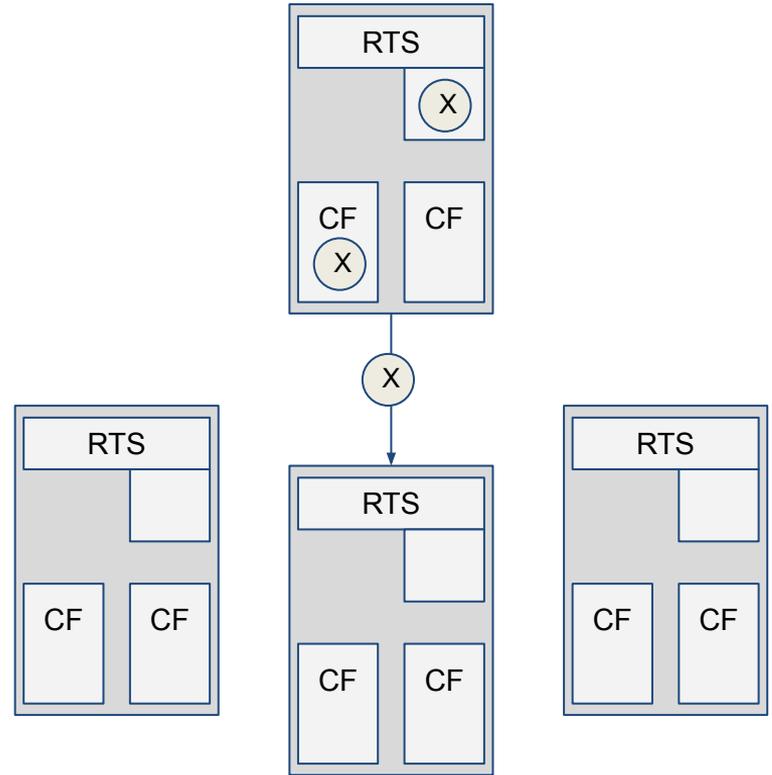
Массовая репликация

1. Фрагмент вычислений обращается к runtime-системе на своем узле.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его всем остальным узлам.
3. Фрагменты вычислений запрашивают у своих узлов нужный фрагмент данных.
4. Как только узел получит требующийся фрагмент данных, он передаст его ожидающим фрагментам вычислений.



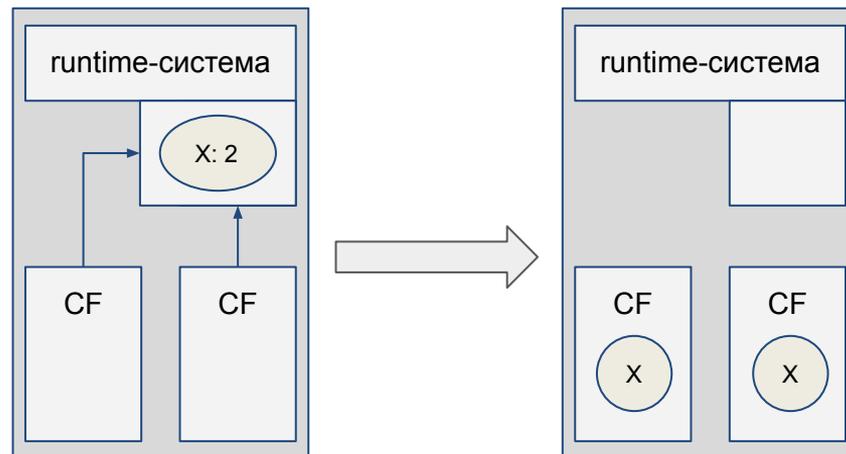
Ограниченная репликация

1. Фрагмент вычислений обращается к runtime-системе на своем узле с данными о степени репликации.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его соответствующим узлам.
3. Фрагменты вычислений запрашивают у своих узлов нужный фрагмент данных.
4. Узел передает данные фрагменту вычислений или передает запрос на фрагмент другим узлам.



Кэширование

1. Фрагмент вычислений обращается к runtime-системе на своем узле с информацией о “длительности хранения”.
2. Runtime-система сохраняет в отдельном контейнере полученный фрагмент данных и рассылает его всем остальным узлам.
3. При каждом запросе происходит проверка на оставшееся количество использований фрагмента.
4. Когда “длительность хранения” фрагмента подходит к концу, он удаляется.



Тестирование

Для тестирования реализованных алгоритмов был предложен тест, который бы использовал фрагмент данных сразу на большом количестве фрагментов вычислений. Были подготовлены реализации теста с использованием стандартных механизмов системы LuNA, массовой репликации и ограниченной репликации.

```
#!/user/bin/luna
import c_init(int, name) as init;
import c_check(int) as check;

sub main()
{
    df x;

    cf a: init(1, x) @ {
        locator_cyclic: 1;
        all_push x;
    };

    for i=2..100000 {
        check(x) @ {
            locator_cyclic: i;
            all_wait x;
        };
    }
}
```

Результаты

Тестирование проводилось на кластере — суперкомпьютере МВС-10П.

	Без алгоритма	Массовая репликация	Ограниченная репликация (половина узлов)
1 процесс	13.028 с	11.913 с	13.208 с
8 процессов	14.460 с	12.777 с	14.125 с

По результатам видно, что скорость программы падает с ростом количества процессов из-за увеличившихся затрат на коммуникации. Как и предполагалось, применение частных алгоритмов массовой и ограниченной репликации позволило сократить время работы программы.

Заключение

В результате выполнения работы были предложены частные алгоритмы управления распределенными данными. Алгоритмы были реализованы в runtime-системе LuNA. Часть реализованных алгоритмов была протестированы.

Защищаемые положения:

1. Разработаны частные алгоритмы и механизмы массовой репликации, кэширования и ограниченной репликации.
2. Массовая репликация, кэширование и ограниченная репликация были реализованы в runtime-системе LuNA.
3. Массовая репликация была реализована в компиляторе системы LuNA.
4. Проведено исследование эффективности программ, использующих частные алгоритмы.

Публикации

- Масыч М. А. Разработка и реализация частных алгоритмов управления распределенными данными в системе LuNA // МНСК-2021. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. Материалы 59-й Международной научной студенческой конференции. С. 116.