Руководство для начинающих по параллельному запуску команд в Linux

Linux Code / 27 декабря 2023 г.

Параллельная обработка – это ключевой метод, позволяющий современным компьютерам выполнять несколько задач одновременно. Используя параллельное выполнение, мы можем значительно ускорить рабочие процессы, эффективнее использовать аппаратные ресурсы и упростить сложную обработку команд.

В этом всеобъемлющем руководстве мы предоставим новичкам Linux дружелюбное введение в параллельный запуск команд. Вы узнаете основные методы параллелизации в Linux, лучшие практики управления параллельными рабочими нагрузками и когда использовать параллельные команды для оптимальной производительности.

Что такое параллельная обработка и почему она важна?

Современные компьютеры оснащены многоядерными процессорами и передовыми операционными системами, которые обеспечивают параллельную работу – возможность выполнять несколько вычислительных задач одновременно.

Параллельная обработка использует эти возможности, разбивая программы и команды на более мелкие части, которые могут выполняться одновременно на ядрах ЦП и системных ресурсах. Это позволяет выполнять работу быстрее по сравнению со строгим последовательным выполнением.

Концепция параллельных вычислений возникла в 1960-х годах с суперкомпьютерами, которые объединяли в сеть несколько процессоров. Но даже стандартные настольные компьютеры, ноутбуки и смартфоны теперь включают параллельную обработку с многоядерными, многопоточными процессорами.

Некоторые ключевые преимущества параллельного выполнения включают в себя:

- **Более быстрая обработка** – распределение рабочих нагрузок по ядрам/системам повышает скорость

- **Более эффективное использование оборудования** – загрузка ЦП и ресурсов сокращает потери
- **Асинхронные рабочие процессы** – действия могут выполняться независимо, без блокировки.

По данным Intel, параллельное выполнение может удвоить производительность на двухъядерных системах. С дополнительными ядрами прирост производительности продолжает масштабироваться.

Для вычислительно-интенсивных рабочих нагрузок, таких как обработка видео, научные вычисления и разработка продуктов, использование параллелизма имеет жизненно важное значение для достижения высокой производительности. Но даже в стандартных системах Linux параллельные команды могут ускорить рутинные задачи по написанию скриптов и автоматизации рабочих процессов.

Далее мы познакомим вас с основными методами выполнения параллельных команд в Linux.

Параллельный запуск команд с точками с запятой

Самый простой способ параллельного выполнения команд Linux – разделение каждой команды точкой с запятой (;).

Например:

```
$ command1 ; command2 ; command3
```

Оболочка будет выполнять каждую командную строку независимо, не дожидаясь завершения предыдущей.

Давайте попробуем простой пример на терминале Linux:

```
$ whoami ; pwd ; ls
user
/home/user
file1 file2
```

Обратите внимание, как вывод `whoami`, `pwd`, и `ls`появился одновременно, а не последовательно. Это показывает, что они выполнялись одновременно параллельно.

Мы можем продолжить цепочку дополнительных команд, используя точки с запятой:

```
$ date ; uptime ; free ; df ; pwd
Sun Jan 1 12:00:00 EST 2023
up 2 days, 10:51
# Memory usage
             total       used       free     shared    buffers     cache
Mem:       2048000    1024000    1024000        800      24000     12800
# Disk usage
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda       41276364 15558252  25718112  38% /
/home/user
```

Оболочка эффективно планирует и чередует выполнение всех команд, выводя результаты по завершении каждой из них.

Несколько советов по использованию параллелизации с точкой с запятой:

- Заканчивайте каждую полную команду точкой с запятой (;)
- Избегайте конвейеризации между параллельными командами
- Следите за выходными конфликтами, если они не перенаправлены.
- Зависимости разума и порядок команд

Точки с запятой обеспечивают простой метод для базовой параллелизации. Но для сложных рабочих процессов необходим больший контроль.

Использование скриптов Bash для параллельных процессов

Для более гибкой настройки параллельной обработки мы можем создать bash-скрипты, запускающие процессы в фоновом режиме.

Сначала напишите скрипт с командами для одновременного выполнения:

```
#!/bin/bash

# Command 1
ping 8.8.8.8 &

# Command 2
find /home -type f &

# Command 3
grep README /apps &
```

Синтаксис &предписывает каждой команде выполняться асинхронно в подоболочке, позволяя следующей запуститься немедленно.

Теперь запустите скрипт, `bash script.sh`и все 3 команды будут созданы одновременно.

`wait`Для синхронизации процессов и управления параллелизмом мы можем использовать :

```
ping 8.8.8.8 &
PID1=$!

grep README /home &
PID2=$!

wait $PID1
wait $PID2

echo "Finished parallel commands"
```

`wait`приостанавливает выполнение скрипта до завершения фонового процесса.

Преимущества скриптов bash для распараллеливания:

- Больше контроля над порядком и координацией
- Многоразовые скрипты для общих рабочих процессов
- Инструменты, такие как `wait`и `$!`для управления процессами

Следите за проблемами, такими как конкуренция ресурсов с чрезмерным параллелизмом. Протестируйте, чтобы найти идеальный уровень параллелизма.

GNU Parallel для расширенного параллельного выполнения

Для более надежной параллельной обработки GNU Parallel предоставляет мощную среду, упрощающую одновременное выполнение команд.

Некоторые примеры использования `parallel`:

```
# Process 4 logs simultaneously
parallel -j 4 process_log {} ::: log_{1..10}.txt

# Unrar archives in parallel
parallel unrar x {} ::: *.rar
```

```
# 6 parallel SCP copies
parallel scp {} server ::: file_{1..6}
```

Основные возможности GNU Parallel:

- Укажите количество одновременных заданий с помощью-j
- Простые аргументы через:::
- Простое управление выходами и кодами возврата
- Занимается координацией и блокировкой

Parallel помогает избежать многих ловушек ручной синхронизации и планирования. Идеально подходит для сложных рабочих процессов.

Для дополнительной функциональности GNU Parallel может интегрироваться с другими инструментами, такими как:

- **find** – Работа с несколькими файлами
- **xargs** – Передача списков аргументов
- **make** – Параллельная компиляция
- **sql** – Запрос к базам данных

Согласно тестам GNU Parallel , инструмент достигает почти линейного ускорения по узлам и ядрам. Это делает его масштабируемым решением для требовательных рабочих нагрузок.

Сравнение методов выполнения параллельных команд

Мы рассмотрели несколько подходов к распараллеливанию в Linux. Вот сравнение их ключевых отличий:

Метод	Настраивать	Контроль	Умение обращаться	Случаи использования
Точки с запятой	Минимальный	Низкий	Руководство	Простые скрипты
Скрипты Bash	Умеренный	Умеренный	Руководство	Пакетные рабочие процессы
GNU Параллельный	Некоторая установка	Высокий	Автоматический	Сложные работы

- Точки с запятой – самый простой способ добавить базовый параллелизм в скрипты.
- Скрипты Bash позволяют реализовывать рабочие процессы и добавляют некоторую синхронизацию.
- GNU Parallel упрощает координацию и идеально подходит для сложных случаев использования.

При выборе подхода учитывайте уровень сложности и необходимость координации.

Когда следует использовать параллелизм для достижения оптимальной производительности?

В целом, параллелизация помогает больше всего с рабочими нагрузками, связанными с вводом-выводом, где время тратится на доступ к дискам или сетям. Преимущества обычно меньше для вычислительно-интенсивных процессов, связанных с ЦП.

Вот несколько примеров, когда выполнение параллельных команд повышает скорость:

- Обработка больших наборов данных или коллекций файлов
- Массовая обработка изображений, кодирование видео, сжатие данных
- Веб-скрапинг, загрузка или выгрузка нескольких файлов
- Перебор хэшей или паролей путем перебора множества комбинаций
- Научное моделирование и математические вычисления

Согласно тестам Кембриджского университета, распараллеливание MATHLAB на 4 ядрах позволило добиться в 3,6 раза более высокой производительности в тестах с плавающей точкой.

Однако параллелизм не панацея – он может вызвать проблемы, если его использовать слишком часто. Рассмотрим закон Амдаля, который показывает пределы ускорения от параллелизации.

Некоторые случаи, когда параллельная обработка может оказаться бесполезной или ухудшить производительность:

- Цепочка операций, требующая последовательных шагов
- Команды, которые активно конкурируют за ресурсы, такие как ЦП, дисковый ввод-вывод
- Работы, требующие больших затрат на координацию
- Приложения реального времени с ограничениями по времени
- Транзакционные рабочие процессы, требующие атомарности

Инструменты бенчмаркинга, такие `time`как профилирование, могут помочь проанализировать потенциальные выгоды. Параллелизм следует тщательно тестировать для каждой рабочей нагрузки.

Заключение и дальнейшие шаги

Мы представили основы выполнения параллельных команд в Linux для ускорения системных рабочих процессов и использования аппаратных ресурсов. Простые методы, такие как точки с запятой и скрипты, могут обеспечить ускорение с минимальными усилиями во многих случаях. Для более сложных рабочих нагрузок GNU Parallel обеспечивает надежную координацию.

С этим вводным руководством у вас должна быть прочная основа для использования методов параллельной обработки для улучшения производительности и продуктивности Linux. Некоторые следующие шаги для продолжения изучения:

- Погрузитесь глубже в передовой опыт и расширенное использование GNU Parallel
- Рассмотрите возможность распараллеливания с другими языками, например, с многопроцессорной обработкой Python.
- Рассмотрите кластерные вычислительные фреймворки, такие как MPI, для многоузлового параллелизма
- Изучите облачные предложения, обеспечивающие параллельные вычислительные мощности

Поиск скорости через параллелизм и параллелизм продолжает развиваться. Спасибо, что присоединились к нам в этом путешествии к более умным и быстрым вычислениям! Дайте нам знать, если у вас есть еще вопросы.

← Предыдущая запись Следующий пост →

Похожие сообщения

«Как проверить историю входов пользователя в систему в Linux?» – полное руководство

Команды Linux

Как системный администратор Linux, вы знаете, насколько важно иметь полную видимость действий пользователей на ваших серверах. Я уверен, что вы…

«Umount Target is Busy» – подробное руководство по устранению этой распространенной проблемы Linux

Команды Linux

Для системного администратора Linux мало что может быть более раздражающим, чем получение ошибки «umount: target is busy» при попытке отсоединить диск или…

100 основных вопросов и ответов для собеседования по Linux

Команды Linux

Linux – это бесплатная операционная система с открытым исходным кодом, широко используемая на серверах, мэйнфреймах, настольных компьютерах, ноутбуках и встроенных системах. Наличие опыта работы с Linux бесценно для…

Более 100 сочетаний клавиш для профессионального использования Linux

Команды Linux

Привет, энтузиаст Linux! Хотите повысить свою производительность и эффективность как опытный пользователь Linux? Освоение сочетаний клавиш – это одно из…

101 команда Linux, которую должен знать каждый пользователь: полное руководство по командной строке Linux

Команды Linux

Интерфейс командной строки Linux (CLI) позволяет создавать мощные скрипты и инструменты для эффективного управления серверами, настольными компьютерами, облачной инфраструктурой и встроенными устройствами. Воспользуйтесь этим…

15 практических примеров Rsync для синхронизации файлов и папок в Linux

Команды Linux

Rsync – это мощная утилита для эффективной передачи и синхронизации файлов между системами. Ее способность передавать только различия и сжимать данные делает ее быстрее…

Поиск...

10 лучших советов по кодированию Linux (по теме),

1. **Параллельное соединение резисторов – полезное пошаговое руководство для начинающих**
2. **Разоблачение SSH: подробное руководство по входу в систему и удаленному запуску команд**
3. **Шпаргалка по основным командам Linux для начинающих**
4. **Выполнение команд CMD в PowerShell: полное руководство 2023 года**
5. **Как выполнить несколько запросов curl параллельно – подробное руководство**
6. **25 лучших команд Linux для новичков**
7. **Как остановить, завершить и управлять длительными командами git log**
8. **Выполнение команд от имени администратора в PowerShell**
9. **Выполнение удаленных команд в Windows с помощью PowerShell Remoting**
10. **Как подключать индукторы последовательно и параллельно: руководство эксперта по Linux**

Недавние Посты

Что такое Oracle APEX? Экспертное руководство по разработке приложений

Надежная обработка ошибок с помощью Try Catch в разработке Linux C#

Руководство эксперта: как получить доступ к режиму восстановления Android

Создание и запуск контейнеров на AWS ECS

Как открыть файлы RAR в Windows 10? Лучшее руководство

Аннотирование снимков экрана в Linux стало проще с Ksnip

Экспертное руководство по устранению бесконечного цикла перезагрузки Windows 10

Установите ONLYOFFICE Desktop Editors, альтернативный офисный пакет для Linux

О

Контакт

политика конфиденциальности

Условия и положения

Отказ от ответственности

Печенье