System Performance Index

System Performance index предоставляет интерфейс для определения производительности и текущей загруженности системы.

Параметры общей производительности позволяют делать вывод о том, может ли компьютер совершать определенную работу, в частности, служить промежуточным агентом для распространения обновлений.

Параметры текущей загрузки могут быть использованы для реализации адаптивного троттлинга. Так, при определенном уровне загруженности системы, менее приоритетный процесс может ограничить свое использование ресурсов в пользу более приоритетных процессов.

Публичный интерфейс един для разных платформ и находится в корневой директории проекта в заголовочном файле SystemPerformanceIndex.h

Интерфейс разделен на две части.

Статические параметры

Статические параметры определяют общую производительность системы. К ним относятся:

- Тип компьютера: ноутбук или стационарный компьютер
- Размер оперативной памяти
- Скорость операций с диском
- Частота центрального процессора
- Количество ядер центрального процессора
- Размеры кэш-памяти центрального процессора

Динамические параметры

Динамические параметры отражают текущую загруженность системы. К ним относятся:

- Текущая загрузка центрального процессора
- Количество свободной памяти на диске
- Количество доступной оперативной памяти
- Очередь процессов к диску (недоступно для OS X)

Реализация

Ссылка на репозиторий

Программа написана для трех платформ: Windows, OS X, Linux. Для них существует единый интерфейс, реализация которого отличается для каждой из платформ. Сборка автоматизирована с помощью системы CMake.

Для реализации интерфейса на ОС **Windows** используются средства Windows Management Instrumentation (WMI) и Performance Data Helper (PDH).

Для реализации на **OS X** используются системные вызовы из библиотек mach и sys.

Реализация для **Linux** выполнена с использованием системных вызовов библиотеки sys и с помощью информации из sysfs и procfs.

Скорость операций с диском на всех платформах вычисляется с помощью бенчмарка.

Сборка

Для сборки необходим CMake версии не менее 3.8. Необходимые действия для сборки на UNIX платформах:

```
$ cd SystemPerformanceIndex
$ mkdir build && cd build
$ cmake ..
$ make
```

Для Windows:

```
$ cd SystemPerformanceIndex
$ mkdir build && cd build
$ cmake.exe ..
```

Далее можно собрать решение с помощью VisualStudio.

Тестирование

Тестирование программы осуществляется с помощью библиотеки Boost Tests, тесты находятся в отдельном проекте и собираются отдельно от основного проекта. Исходные файлы тестов находятся

в директории tests в директории с исходными файлами для нужной платформы (например, SystemPerformanceIndex/MacOS/tests). На компьютере должны быть установлены библиотеки Boost, путь до корневой директории с Boost необходимо указать в CMakeLists.txt в переменной BOOST_ROOT, например:

```
set(BOOST_ROOT "/usr/local/Cellar/boost/1.65.1/")
```

Необходимые действия для сборки тестов:

```
$ cd SystemPerformanceIndex/MacOS/tests
$ mkdir build && cd build
$ cmake ..
$ make
```

Чтобы запустить тесты, нужно выполнить:

```
$ ./SystemPerformanceIndexTest
```

Эта команда запустит основной набор тестов, проверяющих работоспособность программы.

Для проверки консистентности данных о производительности и загрузке системы существуют скриптовые тесты, расположенные в директории tests/Scripts/ под именами StaticTests и DynamicTests. Они написаны на bash script для Liux и OS X и на PowerShell для Windows. Эти скрипты запускают тестовую сборку программы, находящуюся с ними в одной директории. Программа выводит параметры системы в файлы static-parameters.txt и dynamic-parameters.txt. Далее, в скриптах сравнивается вывод программы с значениями соответствующих параметров, полученными из системных утилит (например, top, df, sysctl для UNIX платформ, Get-WMIObject для Windows). Отчет о сравнении имеет вид:

> дата
>
> модель компьютера и операционная система
>
> (для динамических параметров) время, потребовавшееся для работы SystemPerformanceIndex
>
> результаты сравнения в форме: PASS CPUSpeed: actual 2500 MHz, got 2500 MHz

где actual - это результат, полученный из системных утилит got - результат, полученный SystemPerformanceIndex

Исполнять тесты с помощью скриптов можно как вручную, запустив соответсвующий скрипт, так и с помощью автоматической системы тестирования Boost. Для этого необходимо запустить тесты с

дополнительным параметром:

```
$ ./SystemPerformanceIndexTest --run_test=+system_static_parameters_test/test_values
```

```
$ ./SystemPerformanceIndexTest --run_test=+system_dynamic_parameters_test/test_values
```

Или для тестирования и динамических, и статических параметров можно использовать более короткий вариант:

```
$ ./SystemPerformanceIndexTest --run_test=+*/test_values
```

Нагрузочные тесты

Для более детального исследования поведения показателей загруженности системы написаны нагрузочные тесты. Это скрипты, расположенные в директории tests/Scripts/StressTests в соответствующих разделах (CPU, Memory или Disk). Скрипты запускают процессы, активно потребляющие ресурсы системы (CPULoader производит арифметические операции, MemoryLoader аллоцирует и активно использует память, DiskLoader производит копирование большого файла). При запуске каждого нового такого процесса, скрипт записывает параметры загруженности системы, полученные из SystemPerformanceIndex и системных утилит, в файл отчета. Этот отчет может быть использован для визуальной проверки поведения параметров загруженности системы. Для более наглядного представления результатов тестирования системы под нагрузкой скрипт выводит данные в файл с расширением .data, из которого затем строится с помощью программы gnuplot график зависимости параметра системы от количества потребляющих соответсвующий ресурс процессов.

Исполнять нагрузочные тесты с помощью скриптов можно как вручную, запустив соответсвующий скрипт, так и с помощью автоматической системы тестирования Boost. Для этого необходимо запустить тесты с дополнительным параметром:

```
$ ./SystemPerformanceIndexTest --run_test=+system_dynamic_parameters_test/stress_test
_memory
```

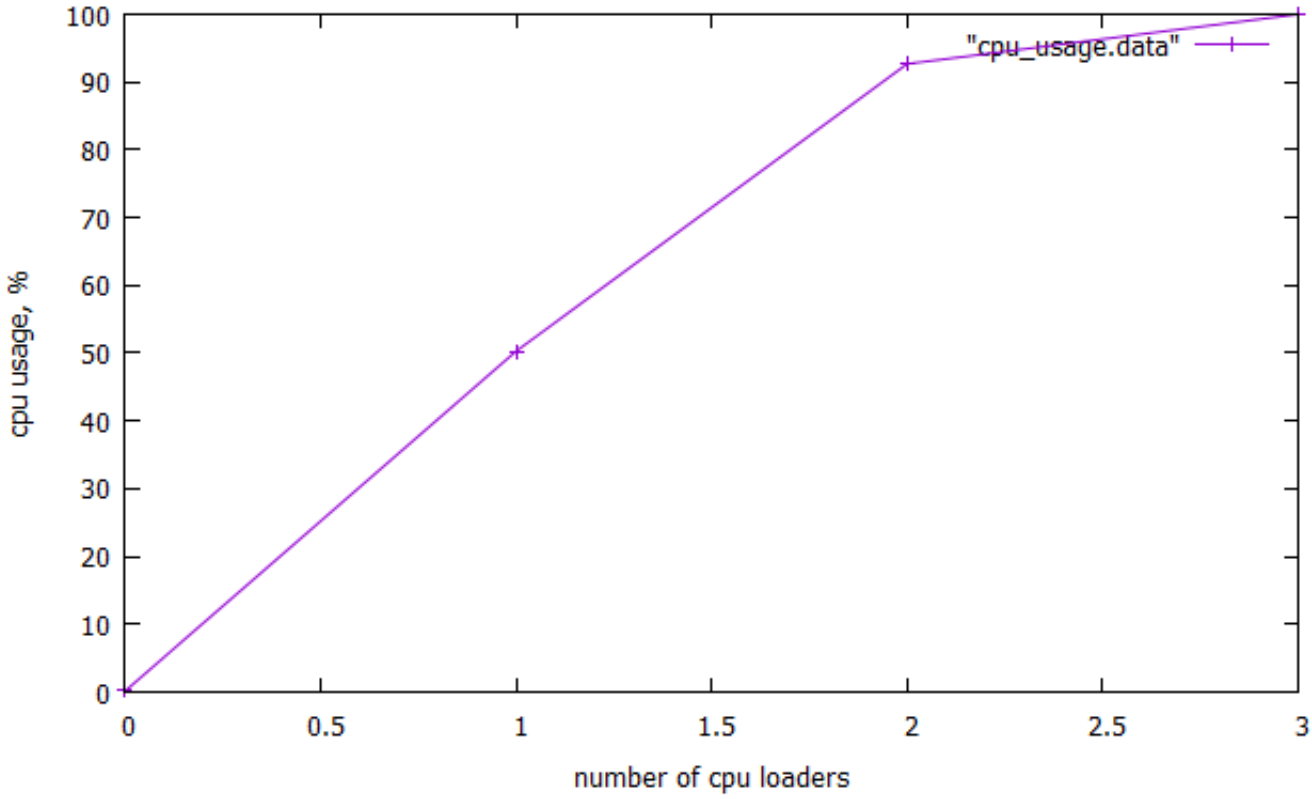
```
$ ./SystemPerformanceIndexTest --run_test=+system_dynamic_parameters_test/stress_test
_cpu
```

```
$ ./SystemPerformanceIndexTest --run_test=+system_dynamic_parameters_test/stress_test
_disk
```

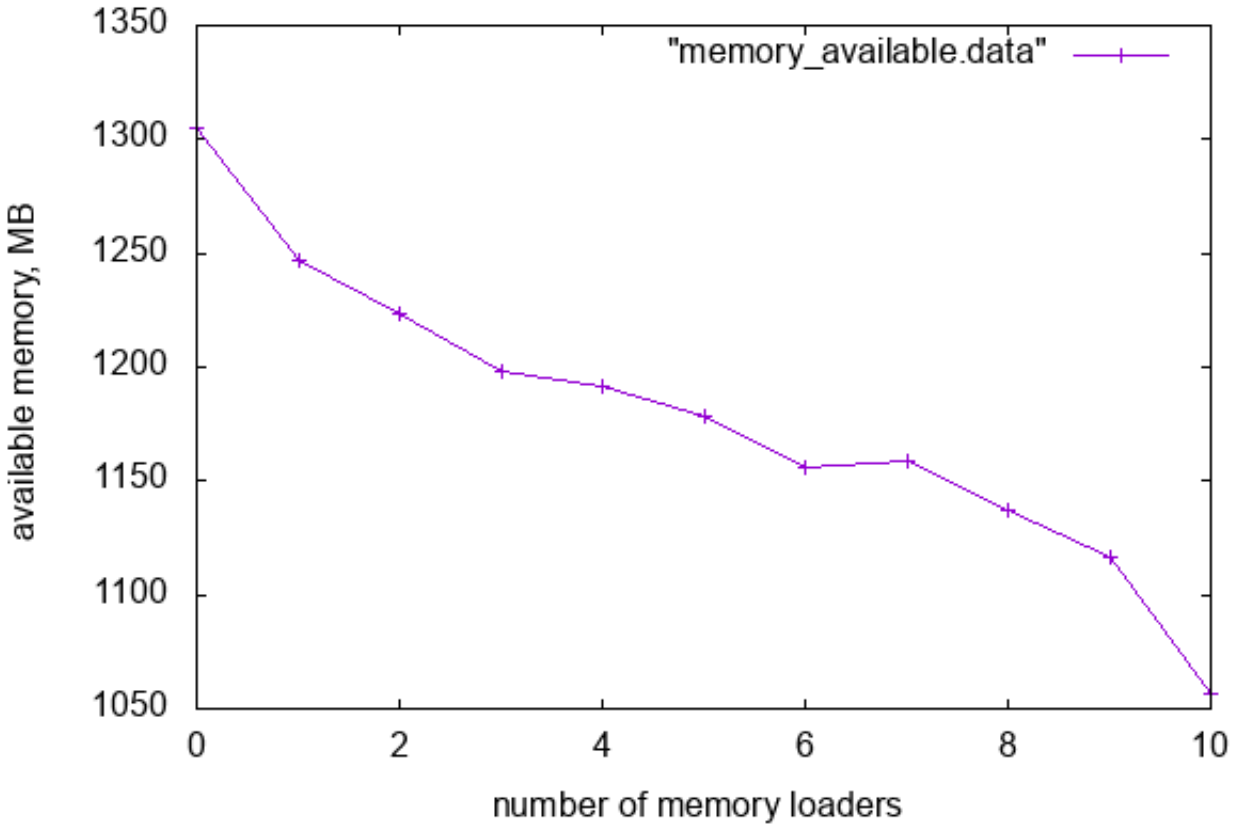
Примеры

Windows

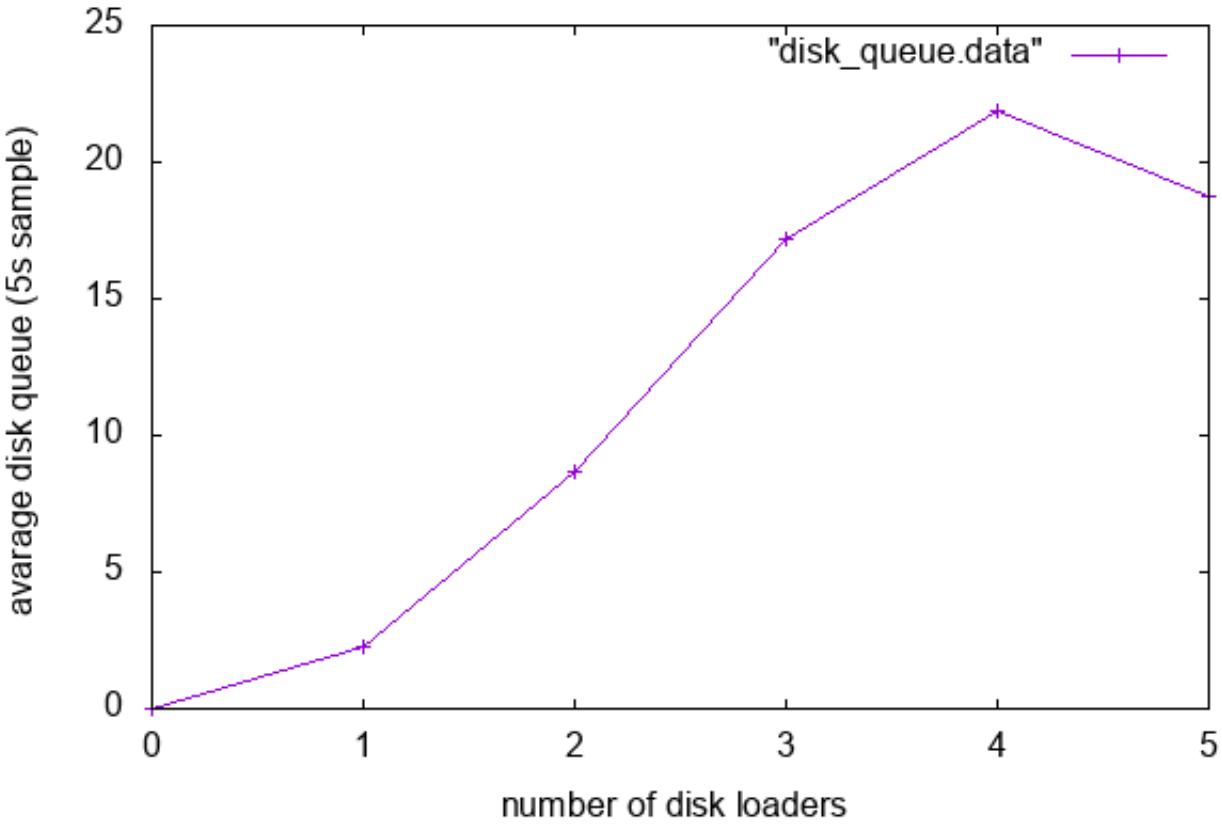
Зависимость загрузки процессора от количества нагрузочных процессов:



Зависимость доступной оперативной памяти от количества запущенных процессов, активно аллоцирущих память:

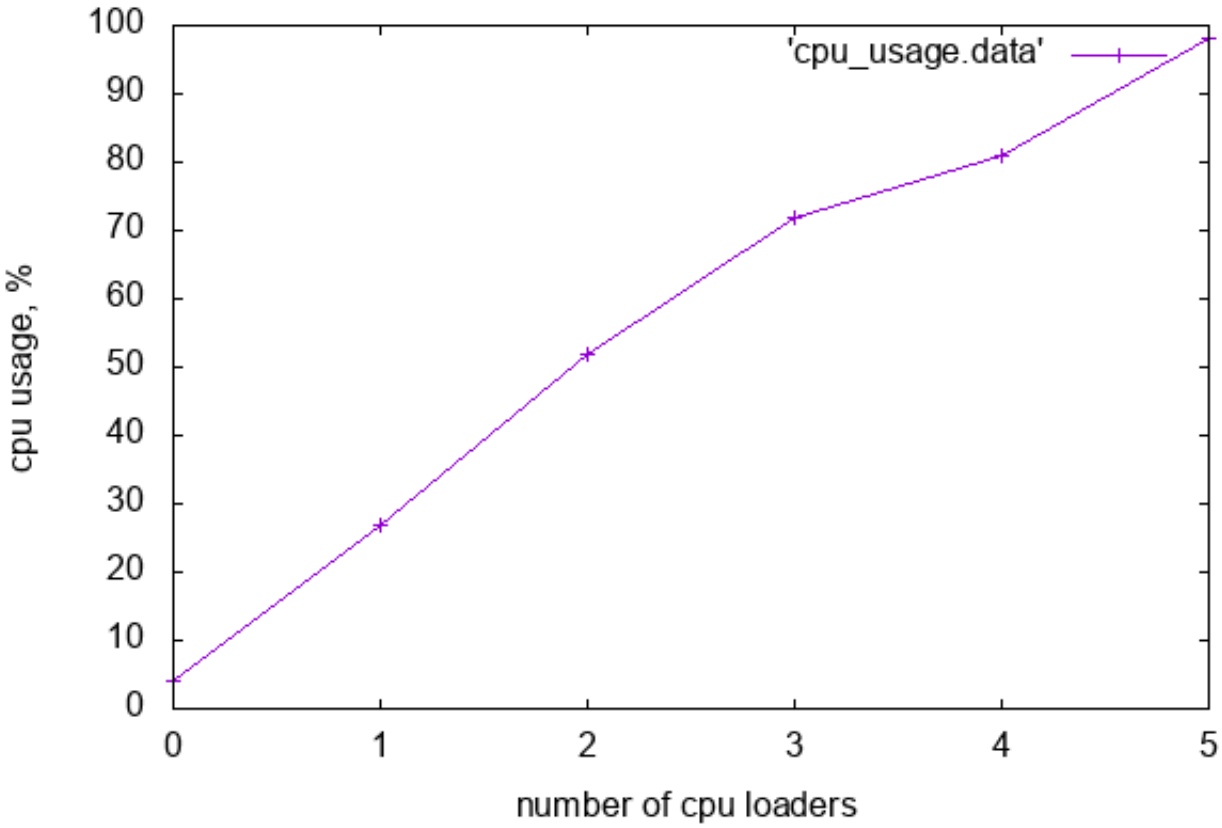


Длина очереди к диску - нестабильный параметр, быстро меняющийся со временем. SystemPerformanceIndex считает усреднение очереди к диску за 5 секунд, что дает более предсказуемый результат, показанный на графике. Зависимость длины очереди к диску от количества запущенных нагрузочных процессов:

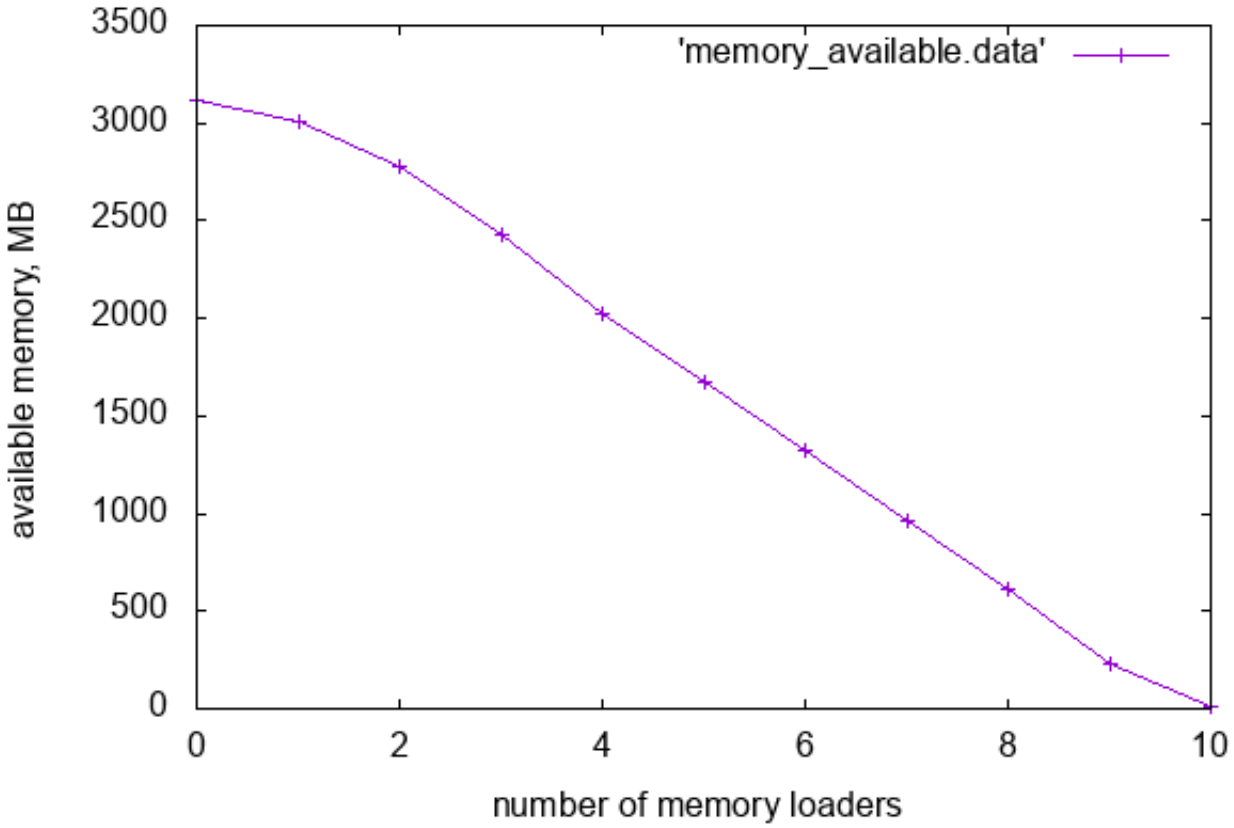


OS X

Зависимость загрузки процессора от количества нагрузочных процессов:

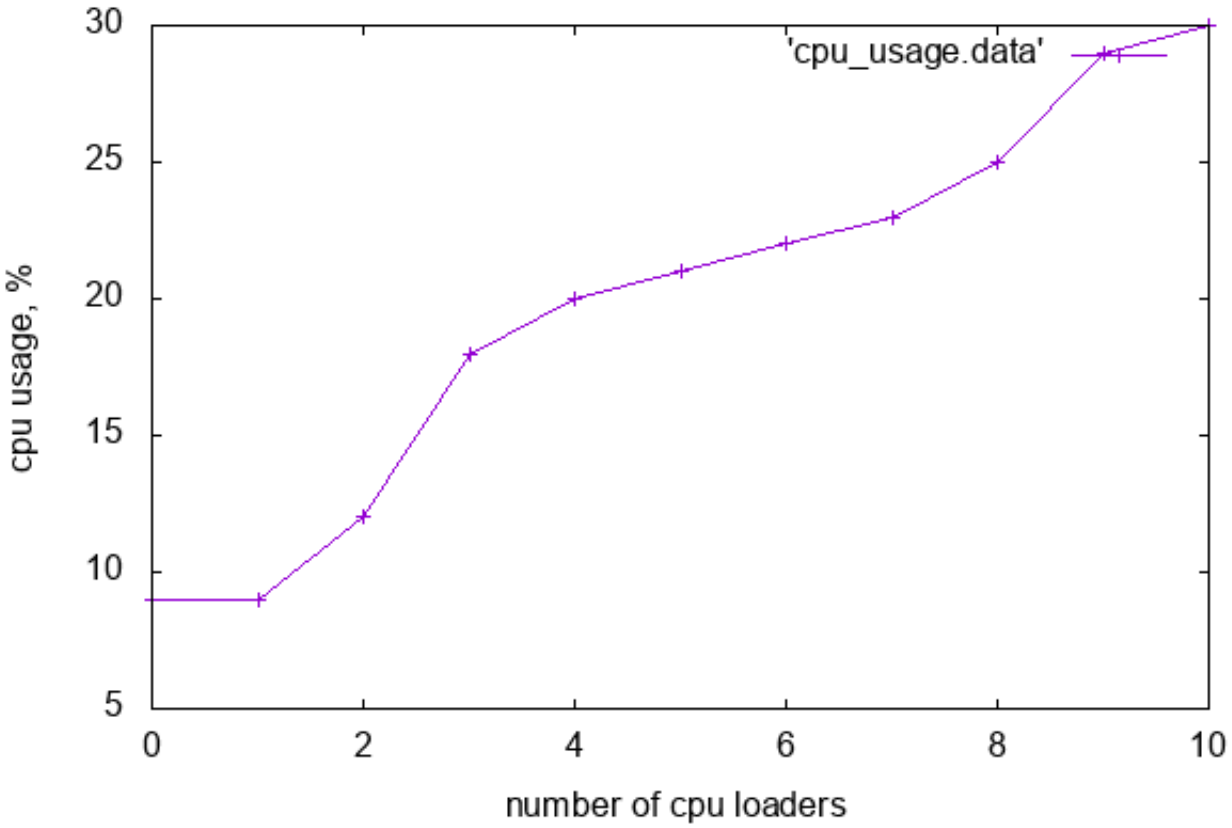


Зависимость доступной оперативной памяти от количества запущенных процессов, активно аллоцирущих память:

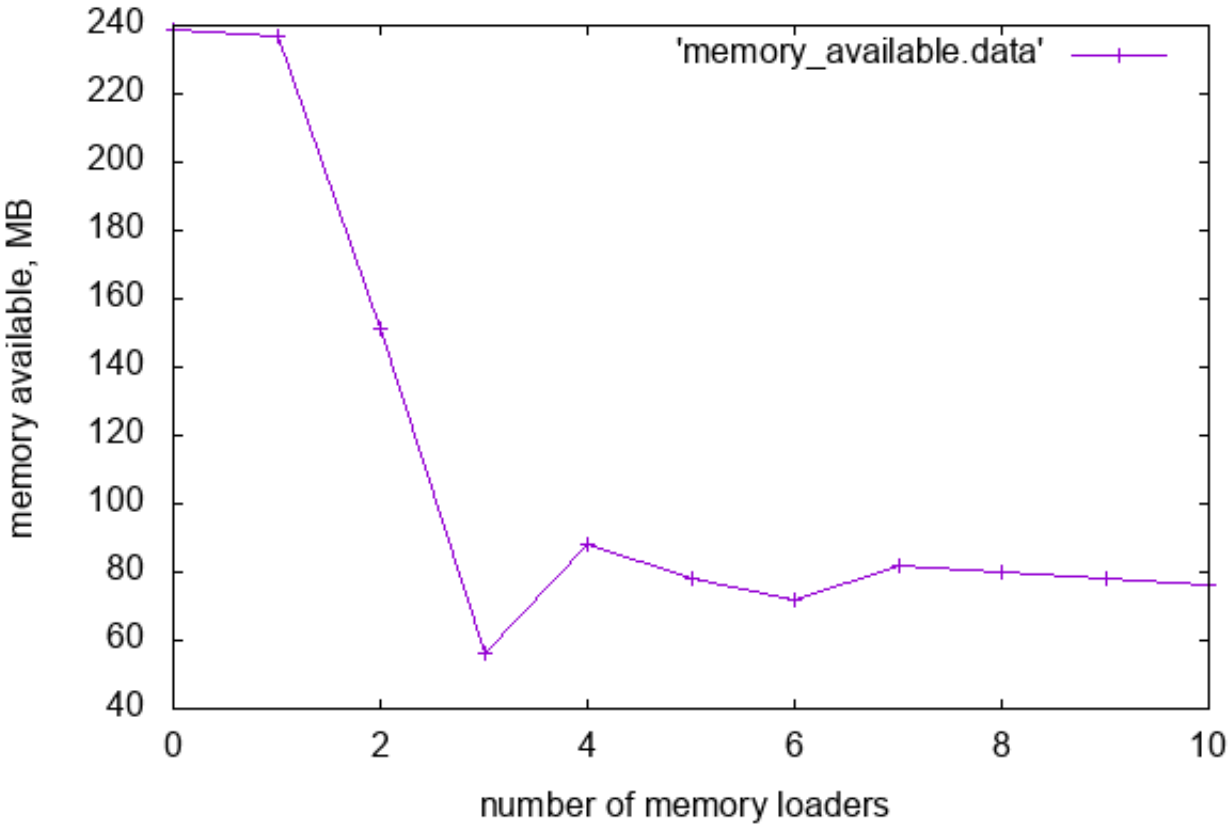


Linux

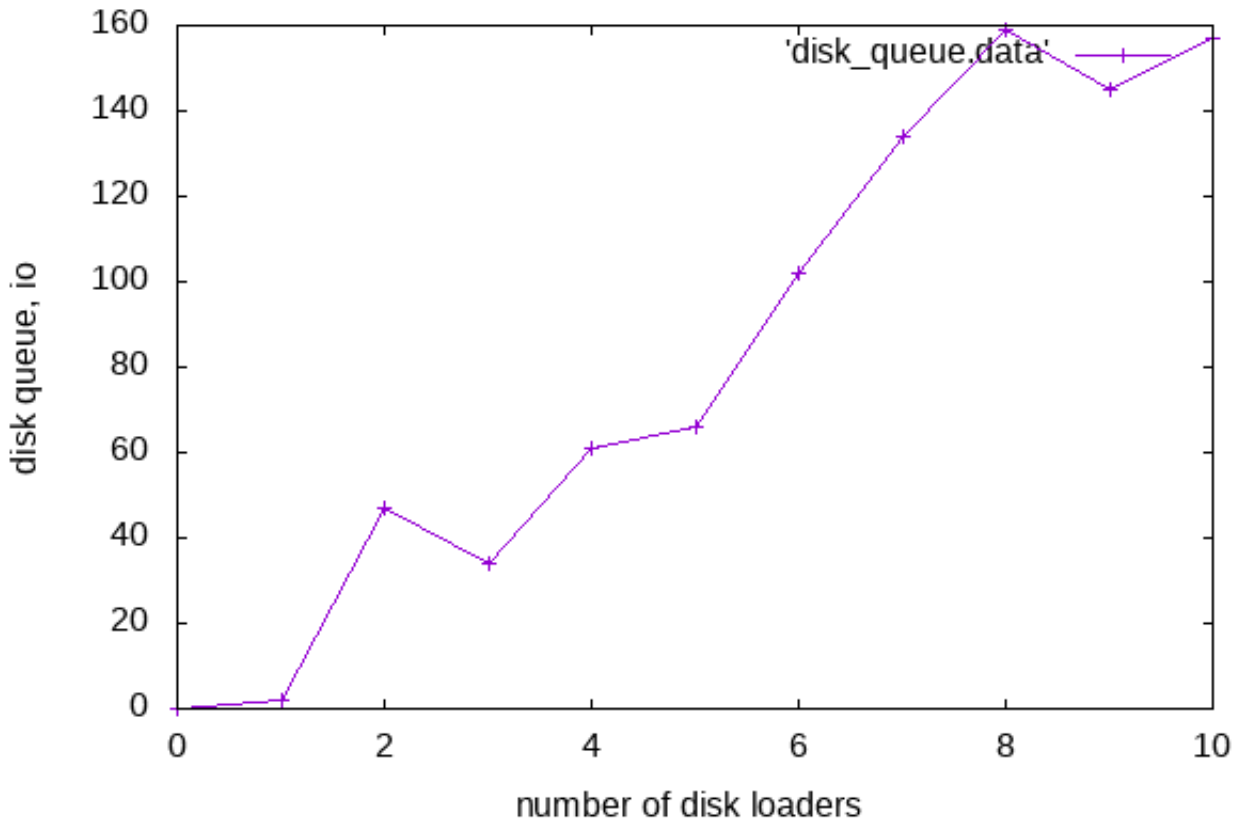
Зависимость загрузки процессора от количества нагрузочных процессов:



Зависимость доступной оперативной памяти от количества запущенных процессов, активно аллоцирущих память:



Зависимость длины очереди к диску от количества запущенных нагрузочных процессов:


'disk_queue.data'
disk queue, io
number of disk loaders
0
20
40
60
80
100
120
140
160
0
2
4
6
8
10