УДК 004.051

ПРОСТАЯ МОДЕЛЬ GRID-СИСТЕМЫ ДЛЯ ОЦЕНКИ ВЛИЯНИЯ АЛГОРИТМА ДИСПЕТЧЕРИЗАЦИИ ЗАДАЧ НА ЭНЕРГОПОТРЕБЛЕНИЕ

И.Л. Кафтанников, К.М. Ханкин

В статье представлена модель состоящей из двух кластеров GRID-системы, созданная в пакете имитационного моделирования AnyLogic и позволяющая оценивать влияние алгоритма диспетчеризации брокера GRID на энергопотребление всего GRID. Модель демонстрирует процесс обработки задач и ведёт учёт среднего энергопотребления на задачу, средней длины локальных очередей, среднего времени ожидания в локальных очередях и плотности распределения среднего времени ожидания. Также описывается алгоритм диспетчеризации, позволяющий снизить энергопотребление GRID, и проводится его сравнение с алгоритмом, выравнивающим длины очередей задач. Описываемый алгоритм распределяет задачи, исходя из наличия или отсутствия на целевом кластере видеоускорителя и наличия или отсутствия реализации задачи для видеоускорителя.

Ключевые слова: grid, энергопотребление, алгоритм диспетчеризации.

Введение

В предыдущих работах авторов была показана необходимость снижения энергопотребления GRID-систем. Основным звеном, влияющим на энергопотребление всей системы, является брокер GRID, отвечающий за маршрутизацию задач. Для оценки влияния алгоритма диспетчеризации на энергопотребление была разработана упрощённая модель в среде имитационного моделирования AnyLogic, позволяющая изменять алгоритм планирования задач, параметры потока задач и объём доступных ресурсов и отслеживать полное энергопотребление системы и в расчёте на одну выполненную задачу.

1. Описание модели

Для моделирования была выбрана среда Anylogic 6.9.0 и её стандартная библиотека. Общий вид модели представлен на рис.1.

Моделируемая GRID-система состоит из двух кластеров, один из которых оснащён вычислительными узлами на базе видеокарт (A), а другой – нет (B). Кластер A содержит два объекта типа ResourcePool, моделирующие свободные процессорные ядра (CPUs_A) и видеопроцессоры (GPUs_A). Кластер B содержит один объект типа ResourcePool, моделирующий свободные процессорные ядра (CPUs_B). Кластеры A и B имеют по 20 процессорных ядер общего назначения, кластер A имеет также 5 видеоадаптеров.

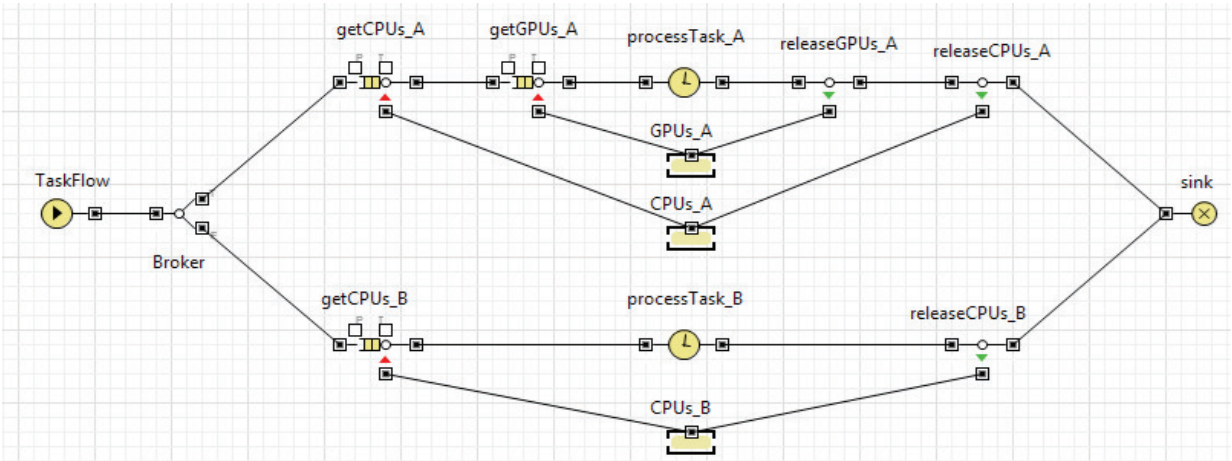

Рис. 1. Общий вид модели

Для описания вычислительной задачи был реализован класс Task, позволяющий определять требуемое для выполнения задачи количество процессоров общего назначения, видеопроцессоров и время на выполнение задачи на требуемом количестве процессоров общего назначения. Новая задача генерируется раз в секунду объектом TaskFlow. При генерации параметры задачи выбираются следующим образом:

1) требуемое количество процессоров общего назначения – от 1 до 10, дискретное равномерное распределение;

2) требуемое количество видеопроцессоров – от 0 до 1, дискретное равномерное распределение (фактически, такое значение параметра говорит о наличии или отсутствии реализации задачи для видеоадаптера);

3) время выполнения задачи – от 1 до 100, дискретное равномерное распределение.

Также в классе Task определено поле, позволяющее флагами задать перечень технологий, используемых для реализации задачи. Анализ этого поля позволит делать выбор узла-исполнителя более гибко, но в данной версии модели этот параметр не используется.

Брокер выполнен как объект SelectOutput с двумя выходами (Broker). Функция выбора выхода и является реализацией исследуемого алгоритма. Для моделирования большего числа кластеров можно использовать либо объект SelectOutput5, имеющий 5 выходов, либо каскадное включение элементов SelectOutput.

При моделировании процесса обработки задач предполагается, что на кластерах используются невытесняющие алгоритмы планирования FIFO. Процессоры захватываются задачами в монопольный доступ. Новые задачи начинают выполняться, как только появляется требуемое количество ресурса для задачи из головы очереди, переупорядочения FIFO не происходит (очередь FIFO может быть превращена в очередь RR с помощью вытеснения задачи по таймауту и помещения её в хвост очереди). Процесс обработки задачи реализован следующими объектами:

1) объекты типа Seize (getCPUs_A, getCPUs_B, getGPUs_A) захватывают требуемое количество ресурса и одновременно является входной очередью бесконечного размера для ожидающих выполнения задач;

2) объекты типа Delay (processTask_A, processTask_B) вносят задержку, моделируя обработку задачи;

3) объекты типа Release (releaseCPUs_A, releaseCPUs_B, releaseGPUs_A) освобождают выделенные на задачу ресурсы.

Для объекта processTask_A введён понижающий коэффициент 10, так как авторами было показано, что скорость обработки задачи, имеющей реализацию для видеопроцессора, до 10 раз выше скорости параллельного исполнения задачи на нескольких процессорах общего назначения [1, 3].

Для учёта энергопотребления были взяты цифры энергопотребления для процессора Core i7 3960X (26 Вт на ядро [2]) и для видеоадаптера nVidia Tesla C2050 (238 Вт на адаптер [3]). Учёт выполняется ежесекундно, по формуле $E = E + (N_{занятых\_процессоров} \times P_{процессора} + N_{занятых\_видеопроцессоров} \times \times P_{видеороцессора} + X)$, где E – энергопотребление в Вт·с, а X – случайная компонента, характеризующая потребление простаивающих ядер и энергозатраты на прочие компоненты кластера. Для экспериментов X формируется дискретным равномерным распределением в диапазоне от 1 до 100. Учитывается как потребление электроэнергии для каждого кластера, так и потребление всей системы, отнесённое к количеству выполненных задач.

Во время работы модели строится два графика. На первом отображается оставшееся на каждом из кластеров время до выполнения задачи, поступившей на исполнительный узел (объект Delay). Этот график позволяет следить за работоспособностью модели. На втором отображается текущее значение показателя энергопотребления системы на выполнение одной задачи. Этот график позволяет определить момент, когда значение показателя может быть принято как достигнутое (момент, когда закончился переходный процесс).

2. Работа модели

Работа модели проверялась на двух алгоритмах диспетчеризации. Первый алгоритм – алгоритм маршрутизации по критерию выравнивания длины очереди на каждом кластере. Второй алгоритм предлагается авторами и заключается в отправке задачи на тот кластер, который способен наиболее быстро обрабатывать задачу, имеющую реализацию с использованием некоей технологии (например, для моделирования взята технология вычислений общего назначения на графическом адаптере).

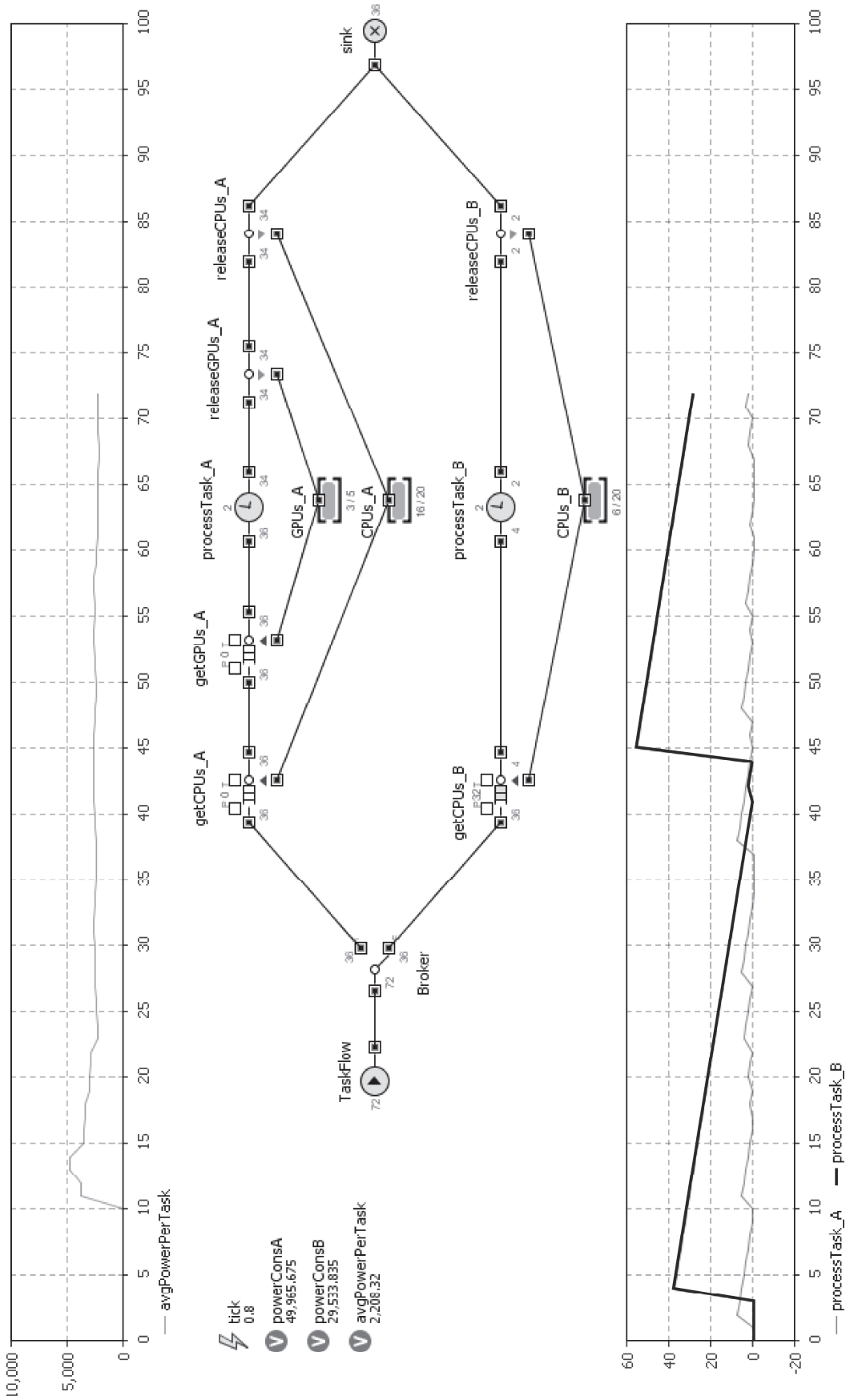

Рис. 2. Переходный процесс, авторский алгоритм

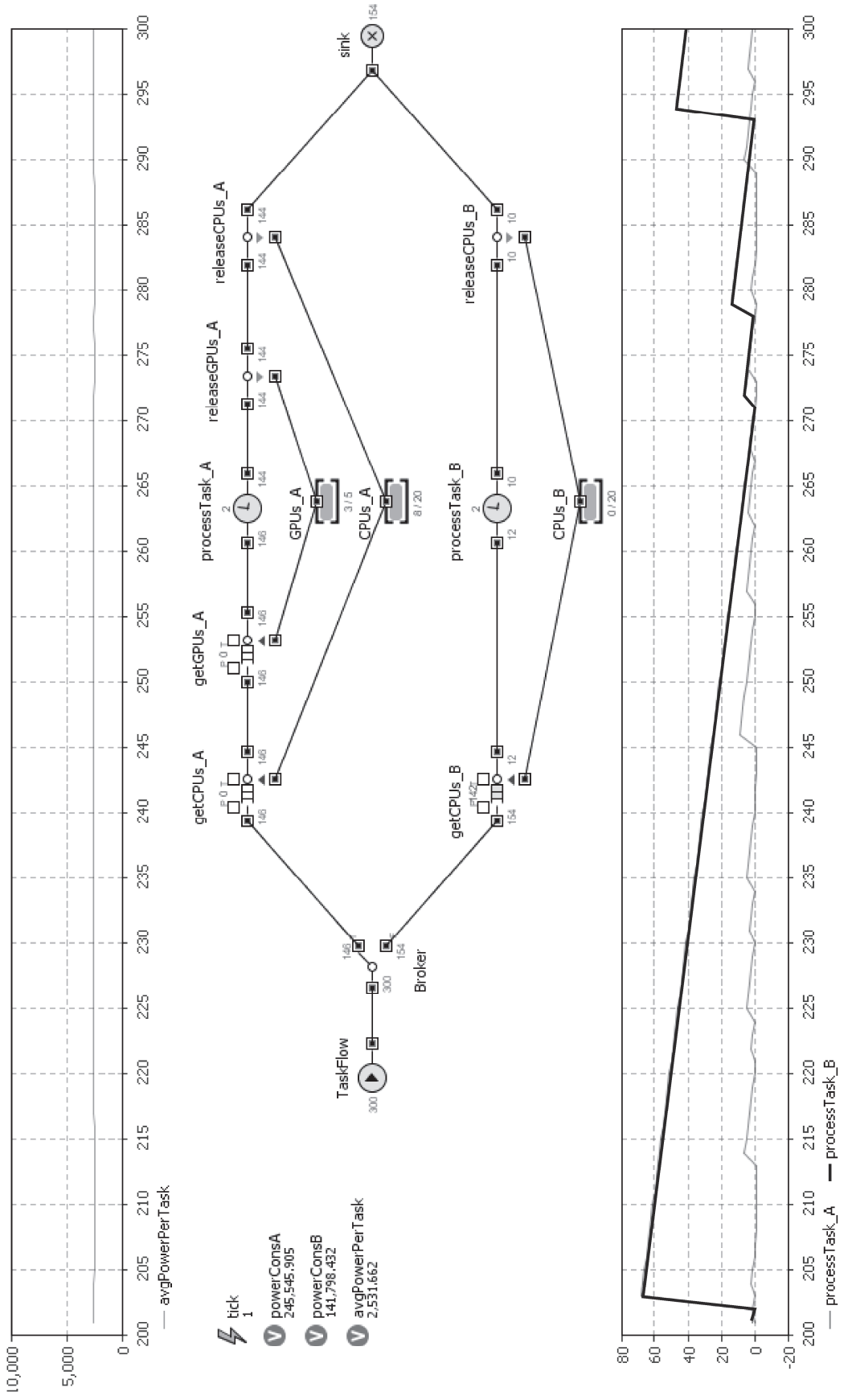

Рис. 3. 300 секунд работы, авторский алгоритм

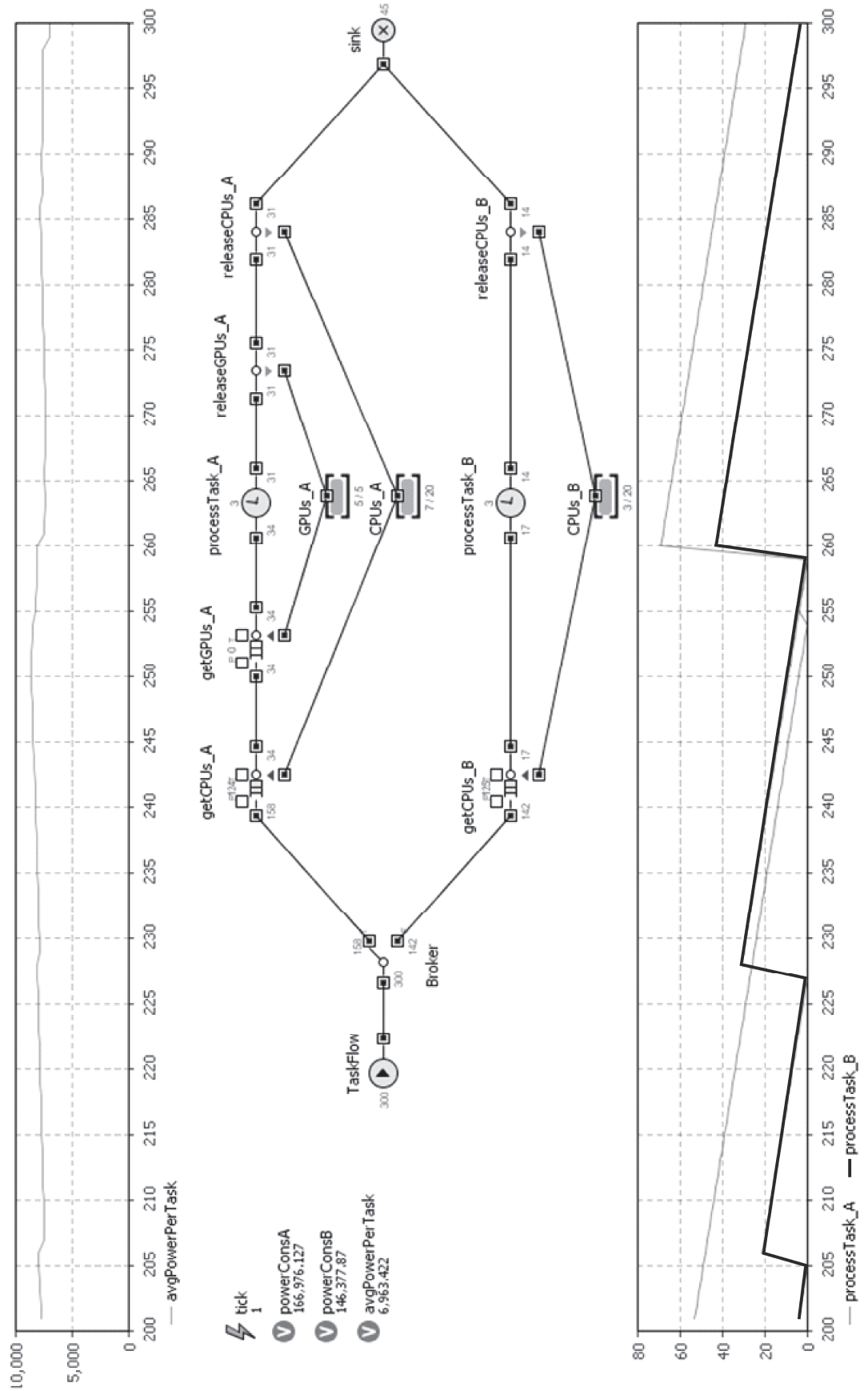

Рис. 4. 300 секунд работы, алгоритм выравнивания очередей

Переходный процесс установления показателя энергопотребления на задачу с авторским алгоритмом представлен на рис. 2. Результат работы модели в течение 300 секунд с авторским алгоритмом представлен на рис. 3. Результат работы модели в течение 300 секунд с алгоритмом выравнивания очередей представлен на рис. 4.

Как видно из графиков, авторский алгоритм показывает в 2 раза лучшие результаты по энергопотреблению. Алгоритм выравнивания очередей на данной модели показывает низкий результат, так как задачи, не имеющие реализации для GPU и, следовательно, выполняющиеся дольше, попадая в голову очереди кластера А, блокируют прохождение задач с реализацией на GPU, выполняющихся быстро, что приводит к нестабильности показателя энергопотребления на задачу, затягиванию переходного процесса и общей неэффективности системы, представленной на модели.

Также видно, что на кластере В, не оснащённом видеоадаптерами, значительно вырос размер очереди задач. Это происходит, так как все задачи, не имеющие реализации для видеоадаптера (а с учётом параметров генерации задач это примерно половина всего числа задач), попадают на кластер В. В силу того, что время обработки в 10 раз выше, чем на кластере А, они образуют длинную очередь.

Выводы

Представленная модель позволяет качественно испытывать различные алгоритмы диспетчеризации, однако нуждается в развитии. Представленный авторами алгоритм в условиях модели показывает в 2 раза большую эффективность по энергопотреблению по сравнению с алгоритмом выравнивания очередей. Негативной стороной предложенного алгоритма является существенная зависимость алгоритма от аппаратного обеспечения составляющих GRID-системы.

Литература

1. Ханкин, К.М. Сравнение эффективности технологий OpenMP, nVidia CUDA и StarPU на примере задачи умножения матриц / К.М. Ханкин // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2013. – Т. 13, № 1. – С. 34–41.

2. Review: Intel Xeon E5-2600 Eight-Core CPUs / Tom's IT Pro. – http://www.tomsitpro.com/articles/xeon-e5-2687w-benchmark-review-cores,2-288-11.html

3. TESLA™ C2050/C2070 GPU Computing Processor. Supercomputing at 1/10th the cost. / NVIDIA Corporation. – www.nvidia.com/docs/IO/43395/ NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

Кафтанников Игорь Леопольдович, канд. техн. наук, доцент кафедры ЭВМ, Южно-Уральский государственный университет (г. Челябинск), kil@comp.susu.ac.ru

Ханкин Константин Михайлович, аспирант кафедры ЭВМ, Южно-Уральский государственный университет (г. Челябинск), hc@comp.susu.ac.ru

Bulletin of the South Ural State University
Series "Computer Technologies, Automatic Control, Radio Electronics"
2013, vol. 13, no. 2, pp. 131–137

A SIMPLE GRID SYSTEM MODEL FOR ESTIMATING A JOB SCHEDULING ALGORITHM IMPACT ON POWER CONSUMPTION

I.L. Kaftannikov, *South Ural State University, Chelyabinsk, Russian Federation, kil@comp.susu.ac.ru*
K.M. Khankin, *South Ural State University, Chelyabinsk, Russian Federation, hc@comp.susu.ac.ru*

The model of GRID system including two clusters is described. Model is created by simulation in AnyLogic. The main purpose of the model is to estimate the impact of

GRID broker scheduling algorithm on GRID power consumption. Model displays task scheduling process and counts average power consumption by task, average local queues length, average waiting time in queues and probability distribution for average waiting time. The scheduling algorithm that makes power consumption lower is described and compared with algorithm of queue length equality. Described algorithm performs scheduling by the criterion of graphic processing unit presence on the cluster and implementation for the GPU presence in task.

Keywords: grid, power consumption, scheduling algorithm.

References

1. Khankin K.M. Efficiency Comparsing of OpenMP, nVidia CUDA and StarPU in the Example of Matrix Multiplication [Sravnenie effektivnosti tekhnologiy OpenMP, nVidia CUDA i StarPU na primere zadachi umnozheniya matrits]. Vestnik YUURGU, 2013, Vol. 13, № 1, pp. 34–41.

2. Review: Intel Xeon E5-2600 Eight-Core CPUs / Tom's IT Pro. – http://www.tomsitpro.com/articles/xeon-e5-2687w-benchmark-review-cores,2-288-11.html

3. TESLA™ C2050 / C2070 GPU Computing Processor. Supercomputing at 1/10th the cost. / NVIDIA Corporation. – www.nvidia.com/docs/IO/43395/ NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

Поступила в редакцию 28 февраля 2013 г.