Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Московский физико-технический институт
(государственный университет)»

Факультет управления и прикладной математики

Кафедра теоретической и прикладной информатики

Предсказание потребления процессорного времени в виртуальных средах

Выпускная квалификационная работа
(магистерская работа)

Направление подготовки: 03.04.01 Прикладные математика и физика

Выполнил:
студент 176 группы — Веринов Александр Алексеевич

Научный руководитель:
кандидат технических наук — Мелехова Анна Леонидовна

Москва

2017

Оглавление

Стр.

Введение

Распространение облачных систем, как публичных, так и внутрикорпоративных, увеличение числа физических машин, находящихся под управлением облачного ПО, вызывает пристальное внимание к системам управления ресурсов – DRS (dynamic resource scheduling). Для оптимального распределения нагрузки DRS должен ориентироваться не только на текущие показатели потребления ресурсов, но и заглядывать в будущее. Прогноз призван дать оценку нагруженности серверов в будущем, на основании чего можно делать выводы о их перенагруженности или недонагруженности.

С одной стороны, оценка потребления ресурсов в будущем важна при запуске новой нагрузке. Тут важно знать, что выбранный для данной нагрузки сервер с большой вероятностью не будет перегружен после добавления нагрузки. С другой стороны, оценка потребления ресурсов нужна при перебалансировке нагрузки. В случае, если некоторый сервер всё же оказался перенагруженным, необходимо мигрировать часть его нагрузки на другие сервера. И выбор нагрузки для миграции, и выбор направления миграции, требуют представление о будущем потреблении ресурсов отдельными нагрузками и их совокупностями.

Конечно, избежать нехватки ресурсов можно оценивая сверху квотой уровень потребления ресурсов, но тогда может остаться неиспользованной значительная часть производительности системы. Таким образом, мы подходим к задаче прогнозирования процессорной нагрузки как влияющей на общую производительность распределённой вычислительной системы.

Предсказание потребления процессорного времени в виртуализованных средах тесно связано с задачей повышения энергоэффективности облачной системы. В этой задаче оценка нагрузки позволяет предельно плотно группировать виртуальные среды на физических машинах, обесточивая неиспользуемые сервера. Своевременное отключение электропитания значительно снижает энергозатраты, которые, согласно ста-

тьям [1–3], составляют существенную часть расходов по поддержанию серверов.

Глава 1. Обзор предметной области

В этой главе будут описаны рассматриваемые нами параметры системы и особенности нагрузок в виртуальных средах.

1.1 Разнородность нагрузки

В виртуальных средах могут выполняться разнородные задачи. Эта разнородность может послужить причиной различного качества предсказания для каждого отдельного метода на различных видах нагрузки. Например, задания, выполняемые на Google Compute Cloud, оказываются более короткими и имеют больший уровень интерактивности, чем обычно выполняемые на grid-системах [4;5]. Более короткие задания приводят к более частым и сильным колебаниям в нагрузке.

Кроме общего характера нагрузки, точность предсказания может зависеть от адаптированности метода к конкретной ситуации. Например, если среди исполняемых в виртуальных средах задач есть взаимозависимые, то учёт возможности такой взаимосвязи открывает перспективы для уточнения предсказания [6–8]. В случае отсутствия такой зависимости, её поиск не может дать выигрыша в точности предсказания, однако вполне может негативно сказаться на вычислительной сложности метода, и ухудшить точность за счёт ложно-положительных срабатываний.

1.2 Параметры для построения прогноза

В подавляющем большинстве статей, посвященных предсказанию процессорной нагрузки [2–21], предсказание основывается на единственном параметре - загруженности процессора (CPU load). В различные моменты времени собираются значения загруженности процессора, фор-

мирующие временной ряд. Однако, это не единственный возможный метод. Например, в работе [1] при использовании кластеризации для повышения точности используется ряд дополнительных параметров: количество исполняемых за цикл процессора инструкций - IPC (Instructions per cycle), количество обращений к памяти - MPC (Memory access per cycle), количество обращений к кэшу - CTPC (Cache transactions per cycle). В мультирегрессионных моделях [22–24] в качестве параметров рассматривались значения всевозможных счётчиков производительности процессора, а также частоты разнообразных системных вызовов. Среди счётчиков производительности особое внимание уделялось IPC, но использовались и счётчики промахов кэша, промахов TLB, операций с плавающей точкой. Не исключено, что эти параметры окажутся полезными и при прогнозировании процессорной нагрузки.

Нагрузка на процессор обычно измеряется в процентном соотношении времени, которое он провел в активном (не idle) состоянии, к общему времени за определенный период (т.е. CPU utilization). Относительная нагрузка - отношение существующей нагрузки к максимально возможной. В статьях, как правило, измеряется относительная нагрузка на физическую машину, представленная как сумма нагрузок, создаваемых всеми выполняемыми на ней заданиями, разделенная на максимальную производительность машины.

1.3 Изучение параметров нагрузки

Прежде чем приступать к разбору методов прогнозирования процессорной нагрузки, сначала следует оценить основные ее параметры, выяснить, какими свойствами обладает временной ряд, по которому строится предсказание.

На этот вопрос довольно подробно отвечает статья [12], в которой изучается реальная процессорная нагрузка на машины в распределенной системе, состоящей из 38 машин, за одну неделю в августе 1997 года.

В этой статье было проведено обширное исследование характеристик временного ряда процессорной нагрузки. Приведем некоторые из них:

- Ряд имел небольшие средние значения при высокой дисперсии.
- Величина дисперсии зависела от среднего значения: чем больше среднее значение, тем больше дисперсия.
- Величина нагрузки имела довольно сложное распределение, однако, в некоторым приближении, ее можно считать нормально распределенной.
- Наблюдалась высокая корреляция значений нагрузки во времени.
- Временной ряд нагрузки не является стационарным. Он проявляет «эпохальное» поведение: нагрузка остается стабильной на определенных отрезках времени, при этом резко изменяясь, когда эти отрезки заканчиваются.

Как уже было сказано во введении, характеристики нагрузки значительно отличаются для различных систем и типов задач. Варьируется изменчивость нагрузки, размер промежутка, на котором ее можно считать постоянной и т.д. Кроме того, конечно же, с 1997 года многое изменилось. Однако статья [12] дает хорошее представление о структуре временного ряда нагрузки в целом. Гораздо более современные исследования характеристик нагрузки проводились на основании данных, собранных Google в своем дата-центре 2011 году [25]. Эта работа также опирается на данные из [25].

Глава 2. Математические сведения

В этой главе даны описание методов прогнозирования, подходов в статистике и машинном обучении, на которые будет опираться дальнейший текст.

2.1 Регрессионные модели

Основной целью регрессионных моделей является предсказание значений зависимых переменных с помощью независимых. Для этого производится поиск функции, которая бы наилучшим образом приближала истинное условное математическое ожидание зависимых переменных при каждом допустимом наборе значений независимых переменных. Эта функция может иметь различный вид. В случае линейной регрессии, предсказываемое значение зависимой переменной равно линейной комбинации зависимых переменных, а обучение состоит в поиске оптимального набора коэффициентов линейной комбинации. Регрессионная модель может строится как на основании некоторого совместного распределения всех переменных, так и без наличия генеративной модели данных.

Для этой цели, в моделях Бокса — Дженкинса временной ряд моделируется белым шумом, пропущенным через линейный фильтр специального вида. Эти модели имеют широкое применение в самых разных областях. Их важной особенностью является последовательный характер. Новое значение ряда моделируется как линейная комбинация прежних значений и прежних ошибок предсказания. Из-за этого, каждое новое предсказание зависит от, вообще говоря, всех прежних значений ряда и ошибок. Такая последовательная природа моделей Бокса — Дженкинса приводит к тому, что, как правило, предсказание осуществляется на продолжении ряда, на котором происходила оценка коэффициентов модели.

Входами регрессионной модели могут быть не только предсказания и истинные значения предсказываемого ряда, но и любые их производ-

ные, например оценка текущей волатильности ряда. Кроме того, оптимизируемая модель сама может рассматривать функции из более шикорого класса, чем исключительно линейные. Увеличение выразительной способности имеет свою цену: растёт риск переобучения, а также теряется возможность проводить статический анализ получаемых моделей. Риск переобучения снижается, если используется большое количество данных. Существуют методы, уменьшающие риск переобучения, такие как регуляризация значений параметров, перекрёстная проверка (cross-validation), метод случайных подпространств, и др.

2.2 Распределения случайных величин

2.2.1 Функция правдоподобия

Функция правдоподобия в математической статистике — это совместное распределение выборки из параметрического распределения, рассматриваемое как функция параметров. Для оптимизации параметров распределений, в работе будет максимизироваться логарифм функции правдоподобия. Метод максимального правдоподобия выбирает такое распределение из параметрического семейства, в котором имеющаяся выборка наиболее вероятна.

2.2.2 Распределение Лапласа

В дальнейшей работе, распределение Лапласа применяется для описания распределений будущих значений потребления процессорного времени. Его плотность изображена на графике 2.1, и задаётся выражением:

$$p(x|\alpha,\mu) = \frac{\alpha}{2}\exp\left(-\alpha|x-\mu|\right)$$

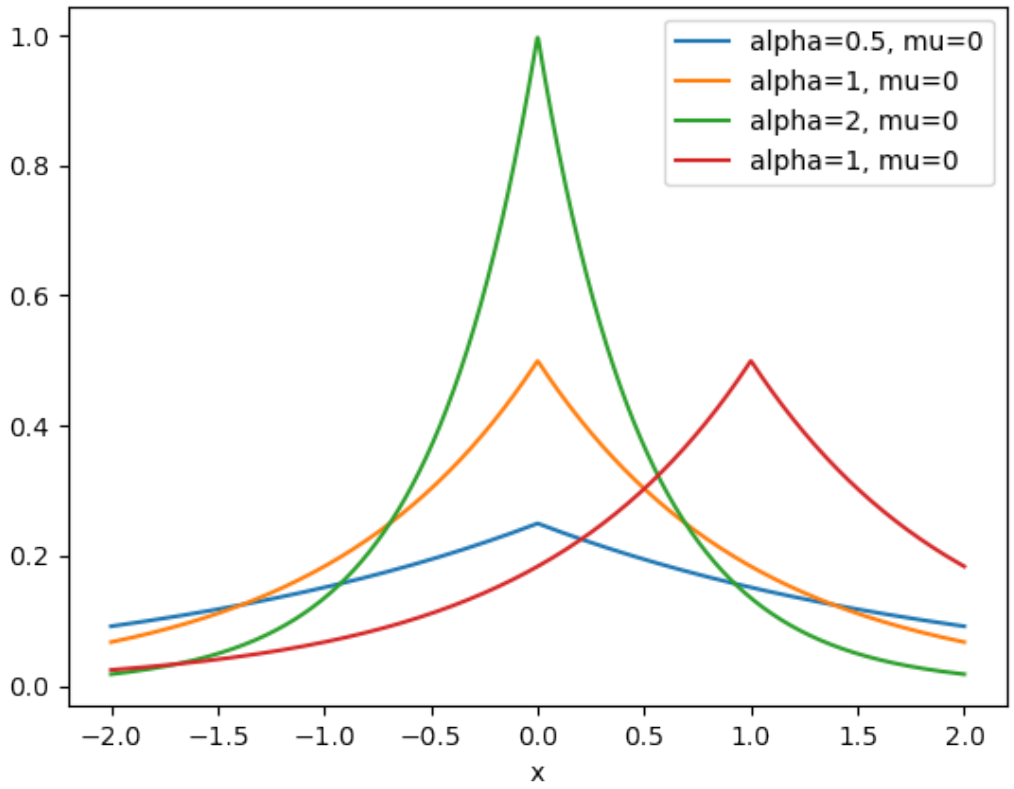

Рисунок 2.1 — Плотность распределения Лапласа

В работе используется усечённая версия распределения Лапласа, в которой реализация заведомо находится на отрезке [0, 1]. После зануления вероятности вне единичного отрезка, функцию плотности необходимо отнормировать. Для этого используется разница функции распределния в точках 1 и 0. Функция распределения:

$$F(x|\alpha,\mu) = \begin{cases} \frac{1}{2}\exp\left(\alpha(x-\mu)\right), & x < \mu \\ 1 - \frac{1}{2}\exp\left(-\alpha(x-\mu)\right), & x \geq \mu \end{cases}$$

2.3 Искусственные нейронные сети

Искусственная нейронная сеть – это вычислительная модель, используемая в машинном обучении. В простейшем варианте модели, ко входным признакам поочерёдно применяются параметризованные линейные функции, и нелинейные функции, называемые функциями активации. Для оптимизации параметрлов линейных преобразований, мини-

мизируется определённая функция потерь. Зачастую, в качестве функции активации применяется tanh.

Как показывается в [26], искусственные нейронные сети могут служить универсальным аппроксиматором для широкого класса функций. Они предоставляют высокую гибкость в определении модели. Фактически, в качестве искусственной нейронной сети можно определить произвольный параметризированный граф вычислений, тем самым задать желаемую структуру модели. Веса искусственной нейронной сети, как правило, оптимизируются с использованием метода обратного распространения ошибки [27].

2.3.1 Различные приёмы

Высокая выразительная способность искусственных нейронных сетей повышает риск переобучения (черезмерно детальное описание тренировочных примеров, с потерей обобщения модели на контрольные данные). Для борьбы с переобучением применяется ряд приёмов.

Один из простых способов борьбы с переобучением является ранняя остановка процесса обучения, когда ошибка на выделенной валидационной выборке начинает расти. Другим распространённым приёмом является затухание весов (weight decay), который на каждой итерации уменьшает значения оптимизируемых параметров. На практике, затухания весов добиваются L_2-штрафом на веса сети [27].

Dropout – весьма мощный метод регуляризации нейронных сетей [28]. Он представляет из себя отдельный слой нейронной сети, который в процессе обучения зануляет долю случайно выбранных входов. Вне обучения, слой масштабирует свой вход, чтобы сохранить ожидаемое среднее значений выхода dropout-слоя, а значит и входа последующего слоя. Как было замечено в [29], dropout можно рассматривать как экономичный способ реализовать ансамбль большого числа нейнонных сетей, работа которых усредняется. Уменьшение взаимозависимостей весов, и

устойчивость ансамбля моделей, позволяют повысить качество предсказаний на шикором классе задач.

Ещё один приём, уже не относящийся к борьбе с переобучением, но существенно ускоряющий обучение: Batch Normalization [30]. Целью этого метода является нормализация входов слоя нейронной сети. Изменение среднего и дисперсии выходов некоторого слоя глубокой нейронной сети приводит к тому, что дальнейшие слои оказываются неприспособленными к получению таких входов, и им требуется дообучение. Этот эффект усиливается с увеличением числа слоём. Если используются функции активации с горизонтальными асимптотами (как tanh), то выходы очередного линейного отображения могут попасть в далёкую от нуля область, где градиент близок к нулю, что может практически заморозить "насыщенный"(saturated) нейрон. Batch Normalization отслеживает среднее и дисперсию входных признаков, нормирует на них проходящие через слой данные, и подбирает среднее и дисперсию производимых выходов (обучаемые параметры слоя).

2.3.2 Вариационные нейронные сети

Суть вариационного подхода заключается в замене точечного предсказания целым распределением. Предсказание параметров оцениваемого распределения можно производить при помощи нейронной сети, что и приводит к вариационным нейронным сетям. Переход от точечных оценок к распределениям влечёт необходимость использовать не оценки точности, вроде среднего квадрата ошибки, а оценки близости вероятностных распределений, вроде расстояния Кульбака – Лейблера.

2.4 Метод Монте-Карло

Вообще говоря, вычисление распределения функции от случайных величин может быть сложной задачей, но существует одна простая и мощная альтернатива. Можно сгенерировать S реализаций случайной величины $x_1, \ldots, x_S$, из них получить реализации рассматриваемой функции $f(x_1), \ldots, f(x_S)$, и на основании эмпирического распределения значения функции делать интересующие вычисления. Методы Монте-Карло широко используются в статистике и машинном обучении [27].

2.4.1 Интергирование методом Монте-Карло

Для примера, можно довольно просто оценить математическое ожидание значения любой функции от случайной величины. Для этого сгенерируем S реализаций x_i, посчитаем среднее арифметическое значений функции на примерах, что и даст оценку математического ожидания. То есть

$$\mathbb{E}\left[f(X)\right] = \int f(x)p(x)\mathrm{d}x \approx \frac{1}{S}\sum_{i=1}^{S} f(x_i)$$

где $x_i \sim p(X)$. Такой приём называется интегрированием методом Монте-Карло. Увеличивая количество используемых реализаций S, теоретически можно произвольно увеличивать точность оценки.

2.4.2 Использование выборки по значимости

Хотя простым интегрированием методом Монте-Карло можно получить произвольную точность оцениваемой статистики, такой подход может потребовать неоправданно много примеров. Например, если оценивается вероятность некоторого крайне маловероятного события, то

столь же малая доля реализаций попадёт в рассматриваемую область вероятностного пространства. В данной работе встретился именно такой случай.

К счастью, в таком случае можно существенно увеличить точно оценки, используя меньшее число реализаций. Идея состоит в генерировании реализаций случайной величины не из изначального распределения, а из некоторого распределения $q(x)$. В идеале, $q(x)$ должно быть пропорционально $f(x)p(x)$, но для корректности метода это не обязательно. Суть выборки по значимости заключена в следующей выкладке:

$$\mathbb{E}\left[f\right] = \int f(x)p(x)\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x \approx \frac{1}{S}\sum_{i=1}^{S} w_i f(x_i)$$

где $x_i \sim p(X)$, $w_i = \frac{p(x_i)}{q(x_i)}$. w_i называются весами значимости (importance weights).

В итоге, усреднее происходит для другой функции с иначе распределённым аргументом. Так как аргумент новой функции распределён уже согласно $q(x)$, вероятность получения значимых для интеграла реализаций X можно увеличить, и тем самым увеличить точность оценки.

Глава 3. Обзор существующих работ

В этой главе рассмотрены результаты работ, изучавших методы предсказания потребления ресурсов в виртуальных средах, в особенности процессорного времени. Приведённые сведения основываются на обзоре [31].

3.1 Регрессионные модели

Результаты, изложенные в работе Dinda от 1999 года [12] и расширенные в его же работе 2000 года, предсказывают процессорную нагрузку на машину (хост) в распределенных системах с помощью моделей Бокса — Дженкинса: AR, MA, ARMA, ARIMA и ARFIMA (см. 2.1). Кроме моделей Бокса — Дженкинса, в статье рассматривались еще три примитивных метода, с которыми производилось сравнение: среднее значение на предыдущем интервале – BM («simple windowed-mean predictor»); среднее как долгосрочное предсказание – MEAN («long-term mean predictor»); и экстраполяция последнего значения – LAST. Точность прогноза определялась среднеквадратичным отклонением.

В работе были получены следующие результаты:

- нагрузка предсказуема («consistently predictable»);
- линейные модели являются достаточно мощными, чтобы предсказывать нагрузку
- MA – модели показали плохие результаты, а остальные модели семейства показали схожие результаты в терминах среднеквадратичных ошибок, поэтому более сложные модели (ARMA, ARIMA, ARFIMA), требующие громоздких вычислений, были отвергнуты
- AR показал себя гораздо лучше, чем BM и LAST
- Накладные расходы в AR(16) пренебрежимо малы

– После всех исследований и тестирований, в статье рекомендуется использовать AR с порядком больше 16

Среди работ, связанных с предсказанием потребления процессорного времени уже отдельными нагрузками, большинство рассматривало разного рода линейные модели. В работе [32] о динамичесом распределении виртуальных машин, была использована авторегрессионная модель второго порядка AR(2). В работе [33], для предсказания была использована линейная регрессия по различным параметрам системы, включающим не только потребление процессорного времени. В работе [34] рассматривались линейная регрессия и однослойная нейронная сеть.

3.2 Байесовский классификатор

Рассмотрим теперь применение байесовской модели для предсказания процессорной нагрузки, как это предложено в статьях [5;10]. Эти исследования демонстрируют, что для Google Compute Cloud метод Байеса имеет высокую точность для долгосрочных предсказаний потребления процессорного времени хостом, и работает лучше других методов, как то: скользящего среднего (moving averages), авторегрессии (auto-regression) и фильтрации (noise filters). Отметим, что сравнивается точность работы классификатора при различных функциях риска, а из параметров временного ряда процессорной нагрузки экспериментально отбираются те, которые дают максимальную точность байесовского метода. Продемонстрировано два метода предсказания – для предсказания средней нагрузки на длительные интервалы времени (от 3 до 16 часов) и для предсказания средней нагрузки на каждом из нескольких последовательных интервалов времени.

Статьи [5;10] работают над тем же набором данных, что и данная работа, однако предсказываемая величина различна. В [5;10], рассматривается потребление процессорного времени физической машиной, на которую планировщик в некотором порядке отправляет задачи на выполнение. Эти нагрузки зачастую имеют небольшую продолжительность

(большинство до получаса). Однако, в исследованиях предсказывается агреригованное потребление, которое скалдывается из множества нагрузок. Каждая из нагрузок имеет свою отдельную продолжительность и время начала. Наша работа фокусируется на предсказании поведения каждой отдельной нагрузки. В частности, для нас остаётся актуальной проблема предсказания момента остановки задачи, когда нагрузка резко становится нулевой. Предсказание для хоста во многом зависит от поведения планировщика задач в кластере, но не от поведения отдельных коротких задач. Из-за этих отличий, в данной работе рассматривается гораздо более короткий горизонт прогнозирования, и, вообще говоря, другие алгоритмы дают высокою точность предсказания.

Тем не менее, часть материалов статей оказалась применимой и в нашем исследовании. В статье [5] вводится 9 признаков (свойств временного ряда нагрузки), характеризующих нагрузку в окне выборки, изучаются их вклады и наиболее эффективные сочетания.

Согласно статье, наиболее важными для предсказания оказались следующие 4 из 9 признаков:

- средняя нагрузка (F_{ml});
- индекс честности (F_{fi}, fairness index) – характеризует степень флуктуации нагрузки. Чем больше индекс честности, тем стабильнее нагрузка: $F_{fi} = \frac{\left(\sum_{i=1}^{d} e_i\right)^2}{d \sum_{i=1}^{d} e_i^2}$
- уровневое состояние (F_{ts}, type state) – пара {кол-во уникальных присутствующих уровней нагрузки; кол-во переходов между уровнями нагрузки}, характеризует разброс значений нагрузки и степень их дрожания;
- первая и последняя нагрузка (F_{fll}, first-last load) – пара {начальный уровень нагрузки; конечный уровень нагрузки}, грубо характеризует тренд.

С некоторыми модификациями, эти признаки были использованы для предсказания.

3.3 Фильтр Калмана

Фильтр применяется для динамического управления ресурсами в многоуровневых (многокомпонентных) серверных приложениях, каждый уровень/компонент которых работает в отдельной виртуальной машине. Критерием эффективности управления является максимизации производительности приложения. Большое внимание уделяется выявлению связей между компонентами приложения, так как при возрастании нагрузки на один компонент, с большой вероятностью она возрастет на связанные с ним. Таким образом, в данных статьях [6;7;17] фильтр Калмана используется для двух основных целей. Во-первых, фильтр используется для отслеживания использования ресурсов (нагрузки) и предсказания их потребления на следующем шаге. Во-вторых, он помогает определить зависимости между компонентами приложения.

Такой подход применяется, когда нужно предсказать поведение системы на один шаг вперед. При этом можно динамически проверять точность предсказания во время исполнения, уточняя предсказанные значения на следующий интервал с учетом ошибки в предыдущем интервале. Фильтр Калмана можно применять для краткосрочных предсказаний, но не для долгосрочных, где он не эффективен [5].

Использование фильтра Калмана в нашем случае осложняется необходимостью учёта вероятности скорой остановки нагрузки. В этих условиях становится неприемлемым предположение о стационарности приращений потребления процессорного времени, и модель перехода системы должна быть более сложной. Кроме того, как отмечалось выше, использование параметров, отличных от самого потребления процессорного времени, улучшает качество предсказаний, а потому может не обойтись без детальных моделей перехода и наблюдения, затрагивающих многие параметры нагрузки и системы.

Глава 4. Симуляции

Для оценки качества алгоритма предсказания потребления процессорного времени была реализована симуляция выполнения нагрузок на физической машине. По задумке, эта симуляция должна повторять существенные черты нагрузок из Google Cluster Data [25]. В частности, симуляция предполагает работу контейнеров, а не гостевых операционных систем.

4.1 Выбор симулируемых аспектов системы

Разрабатываемые алгоритм предсказания предполагается использовать для распределения, в первую очередь, процессорного времени. Поэтому было решено изолировать распределение процессорного времени от распределения иных ресурсов. Детальное моделирование памяти требует большого количества неотносящихся к процессору предположений, например о взаимном влиянии нагрузок. Мы не располагаем данными для информированного ответа на подобные вопросы.

Целью симуляции является проверка и демонстрация возможности применения разрабатываемого алгоритма для планирования процессорного времени в виртуальных средах. Возможными инструментами балансирования нагрузки являются решения о целесообразности миграции из данной физической машины, и о безопасности входящей миграции или запуска новой нагрузки на конкретной физической машине. Такой взгляд приводит к тому, что для оценки качества работы предсказания потребления процессорного времени нет причин моделировать целый кластер. Взаимодействия между многими хостами усложнило бы оценку разумности отдельных решений.

Чтобы изолировать валидацию именно алгоритма предсказания процессорного времени, было решено симулировать только распределение процессорного времени, и только на одной физической машине. Так,

оказывается возможным оценить точность выводов алгоритма о распределении вероятностей уровней потребления процессорного времени в разные моменты в будущем, и возможность их применения. При этом, снимается зависимость и от эффективности работы балансировщика нагрузки, и от потребления нагрузками прочих ресурсов.

4.2 Реализация

Конструкция симуляции весьма проста. Симулируется физическая машина с некоторой производительностью процессора, на неё подаются нагрузки на выполнение. Нагрузки расположены в некотором фиксированном порядке, порядок запуска нагрузок планировщик менять не может. Свобода планировщика ограничена лишь серией бинарными решениями: запустить сейчас очередную нагрузку, или подождать.

Шаг симуляции хоста заключается в следующих этапах:

- запускать новые нагрузки из очереди, пока планировщих это позволяет
- собрать текущие потребления нагрузок и их квоты
- распределить имеющиеся ресурсы между нагрузками, симулируя взвешенную справедливую очередь (она реализована в текущем планировщике выполнения задачи ядра Линукс – Completely Fair Scheduler), веса которой соответствуют квотам по процессорному времени
- используя доли, с которой были удовлетворены потребности нагрузки в процессорном времени, обновить прогрессы выполнения нагрузок
- исключить из рассмотрения завершившиеся нагрузки

В процессе работы симуляции, суммарное потребление ресурсов нагрузками логируется. С этими данными, после симуляции возможно исследовать частоту и степень нехватки процессорного времени, вызванную слишком агресивным запуском новых нагрузок.

При нехватке ресурсов, алгоритм предсказания получает симулируемое потребление процессорного времени, а не историческое. Когда нагрузке не удаётся выделить запрашиваемое процессорное время, её прогресс замедляется. Замедление прогресса нагрузки приводит к увеличению длительности её выполнения, никакой потери прогресса при уменьшении производительности не предусмотренно.

Глава 5. Содержание работы

В этой главе описываются использованные данные и проделанные исследования. Исследовательская часть работы состоит из двух основных частей. Вначале, были опробованы различные алгоритмы предсказания потребления процессорного времени, используя тренировочные данные, и на тестовых данных была оценена их точность. Затем, был разработан алгоритм для предсказания распределения прогнозируемого потребления процессорного времени, который был применён при симуляции планирования нагрузки на сервер.

5.1 Постановка задачи

Существующие работы по предсказанию процессорной нагрузки использовали различные источники данных, и зачастую фокусировались на предсказании потребления для всего сервера. Для балансирования нагрузки на сервера больший интерес представляет предсказание потребления ресурсов отдельными нагрузками, ведь общая нагрузка на сервер уже во многом определяется планировщиком. Поэтому первой задачей является сравнение существующих подходов к предсказанию потребления процессорного времени отдельными нагрузками на едином источнике данных.

Выбранный лучший метод предсказания потребления процессорного времени должен быть практически применим для распределения нагрузок. В частности, для контроля вероятности перегрузки необходимо иметь представление о степени неопределённости предсказания. Отсюда вторая задача: адаптировать алгоритм предсказания для получения не точечного прогноза, а распределения вероятностей, и продемонстрировать применимость прогнозов для эффективного распределения нагрузок. Устройство симулирующей среды и обоснование его выбора изложено в главе 4.

5.2 Использованные данные

В большинстве рассмотренных статей использовались различные эталонные нагрузки. Этот подход имеет свои достоинства, как то: контроль за сбором данных, контроль конфигурации системы, действительное отрабатывание механизмов операционных систем и оборудования. Однако, этот способ сбора данных затратен по времени и ресурсам, и, что более важно, все наблюдаемые тренды явно или неявно порождаются исследователями. Рост или падение нагрузки через минуту определяется лишь заведомо заданными настройками и устройством конкретного теста, для которых нет объективных причин отражать общие тенденции возникающих в действительности нагрузок. В такой среде, безусловно, можно испытывать алгоритмы на характерных и краевых случаях, но не в среднем.

В силу слабой применимости синтетических нагрузок для наших целей, было решено опираться на записи о работе реального кластера. Были использованы логи 29 дней работы кластера компании Google. В них запечатлено исполнение задач на порядка 12.5 тысячах физических машин в мае 2011 года. Этот набор данных нередко используется в публикациях о потреблении ресурсов в виртуальных средах [5; 10; 35–37], из открыто доступных он один из самых крупных и недавних.

Алгоритмы обучались и тестировались на строго разделённых по времени нагрузках. Следуя опыту [5], для обучения были взяты данные из дней с 1 по 25, включительно, тестировались обученные алгоритмы на днях с 26 по 29. Нагрузки, время работы которых пересекало границы выбранных временных интервалов, не участвовали в экспериментах. Первая половина тестовых дней была использована для валидации алгоритмов, а вторая была отложена для итогового сравнения и проверки алгоритмов.

5.2.1 Описание данных

Данные о работе нагрузок записывались, как правило, с периодом в 5 минут. Для каждой нагрузки доступна информация о запрашиваемых ею квотах, её приоритете, пользователе и др.

Для нагрузок в каждый отчёт времени записывались следующие динамические показатели:

- среднее потребление процессорного времени (mean CPU usage rate),
- объём доступной нагрузке памяти (canonical memory usage),
- объём отведённой нагрузке памяти (assigned memory usage),
- объём непривязанных ни к какому процессу страниц кэша (unmapped page cache memory usage),
- объём используемого кэша (total page cache memory usage),
- максимальный за интервал измерения объём доступной нагрузке памяти (maximum memory usage),
- доля времени работы с дисками (mean disk I/O time),
- среднее использование локального дискового пространства (mean local disk space used),
- максимальное за интервал измерения потребление процессорного времени (maximum CPU usage),
- максимальная за интервал измерения доля времени работы с дисками (maximum disk I/O time),
- циклов процессора на инструкцию (cycles per instruction – CPI),
- доступов к памяти на инструкцию (memory accesses per instruction – MAI).

5.2.2 Подготовка данных

Вначале, из логов работы кластера были извлечены нагрузки, полностью лежащие в тренировочном интервале времени, и тестовом интер-

вале времени. Потребления ресурсов нагрузками были отнормированы, так что большинство динамических величин были приведены в общий масштаб. Данные об квотах, использованные для нормировки потребления ресурсов, были сохранены и использованы для предсказаний.

5.2.3 Выбор горизонта предсказания

Выбор желаемого горизонта прогнозирования определяется последующим применением прогноза. Данная работа рассматривает прогнозирование потребления процессорного времени, как инструментальную задачу для распределение вычислительной нагрузки в распределённой системе. Поэтому горизонт прогнозирования должен быть заведомо больше характерного времени, за которое возможно разрешить перегрузку отдельного сервера, и период времени с наибольшей вероятностью перегрузки должен оказаться внутри горизонта прогнозирования.

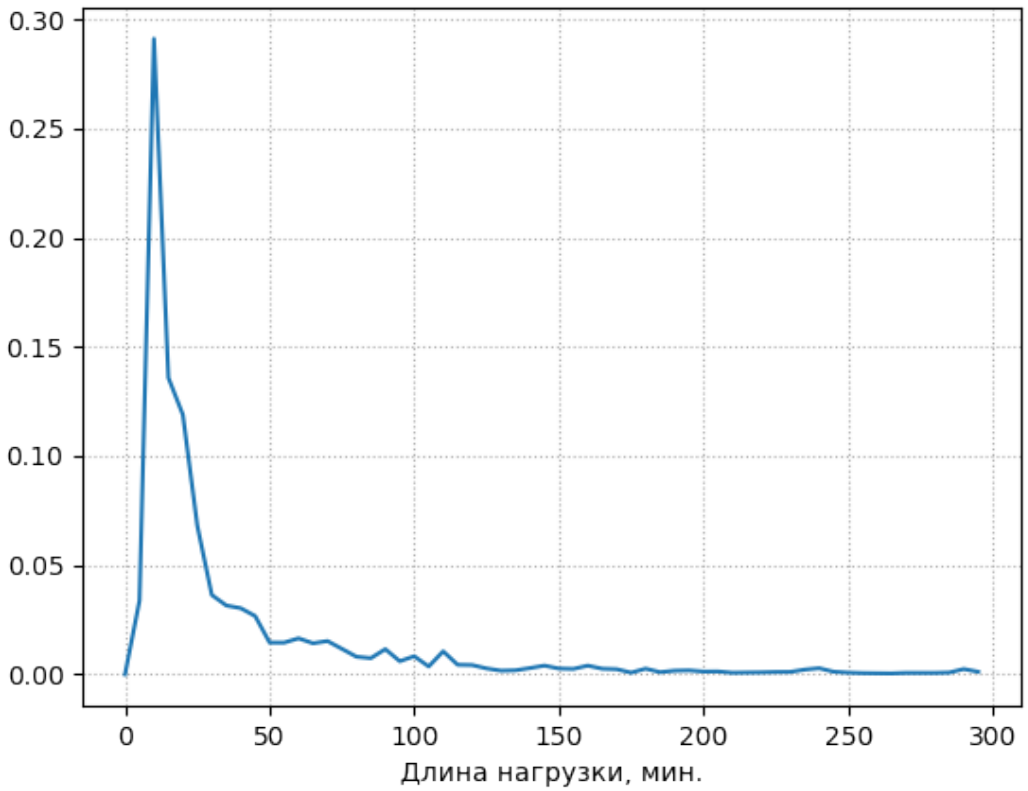

Рисунок 5.1 — Распределение длин нагрузок

На графике 5.1 представлено распределение длин нагрузок в тренировочных данных. Это распределение имеет медиану на 20 минутах (4 отчёта), и среднюю длину 47.2 минуты (9.44 отчёта). Иными словами, большая часть нагрузок, для которых будет прогнозироваться потребление процессорного времени, через 20 минут уже будут завершены. Поэтому было решено зафиксировать горизонт прогнозирования 20 минут.

5.2.4 Выбор признаков

До применения алгоритмов предсказания, стояла задача выбора величин, на основании которых будут производиться предсказания.

Для повышения однородности входов алгоритмов, потребления ресурсов нагрузками были отнормированы на квоту потребления этих ресурсов. Поскольку данные о квотах всё ещё остались в числе входов большинства применённых алгоритмов, информация об истинном масштабе осталась доступной алгоритмам предсказания.

Были взяты:

- статические характеристики нагрузки: запрос по процессорному времени, дисковому времени, памяти
- последние 5 измерений всех динамических величин, дополненные нулями
- среднее за всё время для всех динамических величин
- скользящее среднее всех динамических величин, с $\alpha = 0.5$ и $\alpha = 0.8$
- индекс честности $[5; 10]$ всех динамических величин
- время со старта выполнения

5.3 Сравнение моделей предсказания

Другие работы, освещающие предсказание потребления процессорного времени, зачастую затрагивали потребление процессорного времени другими сущностями (физический сервер, виртуальная машина), либо оценивали точность на синтетических данных, либо же на других исторических данных. Для получения сравнимой оценки точности в выбранной постановке эксперимента, модели из этих работ были заново применены. Большинство исследований использовало те или иные линейные модели, отличающиеся набором входных параметров. В приведённом ниже сравнении приняли участие линейные и нелинейные модели. Линейной моделью была линейная регрессия с регуляризацией Тихонова, нелинейной моделью выступала специально спроектированная трёхслойная нейронная сеть, которая использует все техники, описанные в разделе 2.3.

Алгоритмы предсказания сравнивались на различных наборах входных параметров: последние 5 измерений потребления процессорного времени, последние измерения всех динамических параметров нагрузки, и полный набор всех параметров, приведённых в разделе 5.2.4.

Для сравнения моделей использовался коэффициент детерминации R^2 для каждого из четырёх горизонтов, и его среднее значение по горизонтам. Значение R^2 выражает долю объяснённой дисперсии прогнозируемой величины. Например, $R^2 = 0.9$ означает, что дисперсия ошибки предсказания составляет десятую часть дисперсии предсказываемой величины.

Результаты сравнения приведены на графике 5.2. Как видно, лучший результат был получен при использовании нелинейной модели на всех входных данных. Использование лишь последних измерений динамических величин оказывается недостаточным. Нелинейные модели показывают существенно более высокую точность на дальних горизонтах прогнозирования, чем линейные модели.

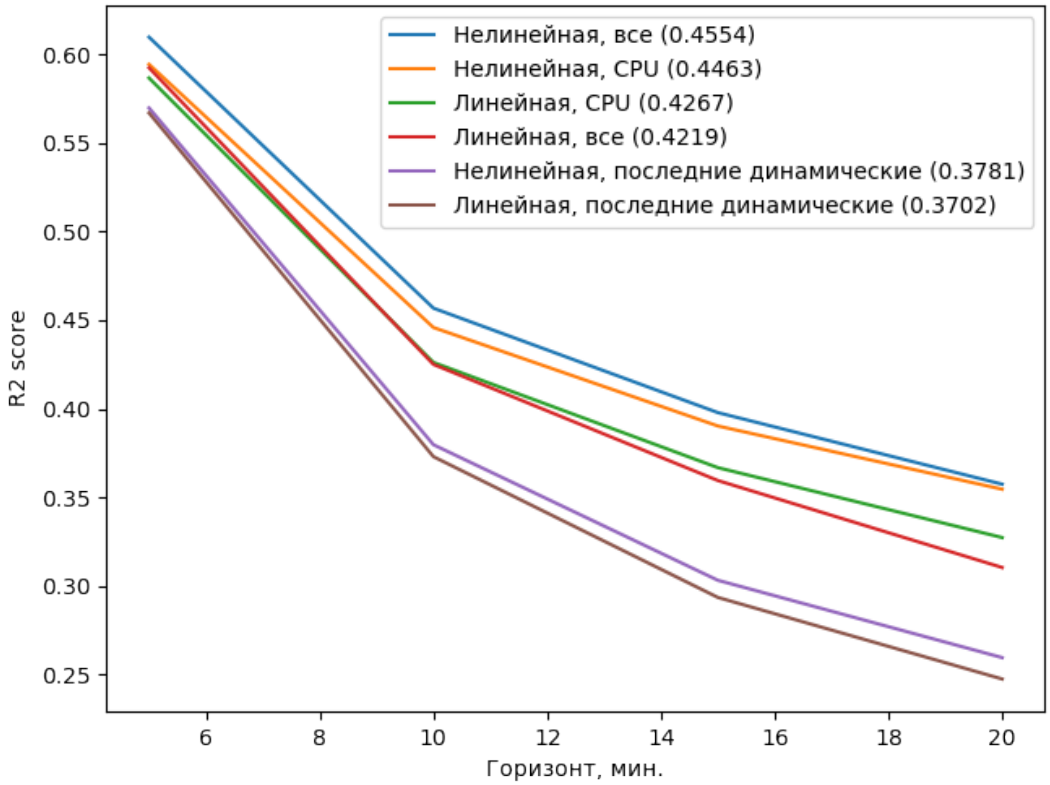

Рисунок 5.2 — Сравнение точности предсказания различных моделей

5.4 Предсказание распределения

Успешное применение искусственной нейронной сети для оценки среднего значения предстоящего потребления процессорного времени необходимо распространить на оценку всего вероятностного распределения.

На тренировочных данных 23% предсказываемых значений равны 0, и 20% значений равны 1. Такая особенность не согласуется с непрерывными распределениями, т.е. истинное распределение имеет непрерывную и дискретную составляющие. Распределение ошибок предсказательной модели на примерах, истинное значение нового потребления процессорного времени которых не равно 0 или 1, имеет характерную для распределения Лапласа форму (рис. 5.3), поэтому для описания непрерывной составлябщей было решено брать именно его.

В качестве алгоритма предсказания были выбраны вариационные нейронные сети (стр. 13). Выходами сети являются параметры пред-

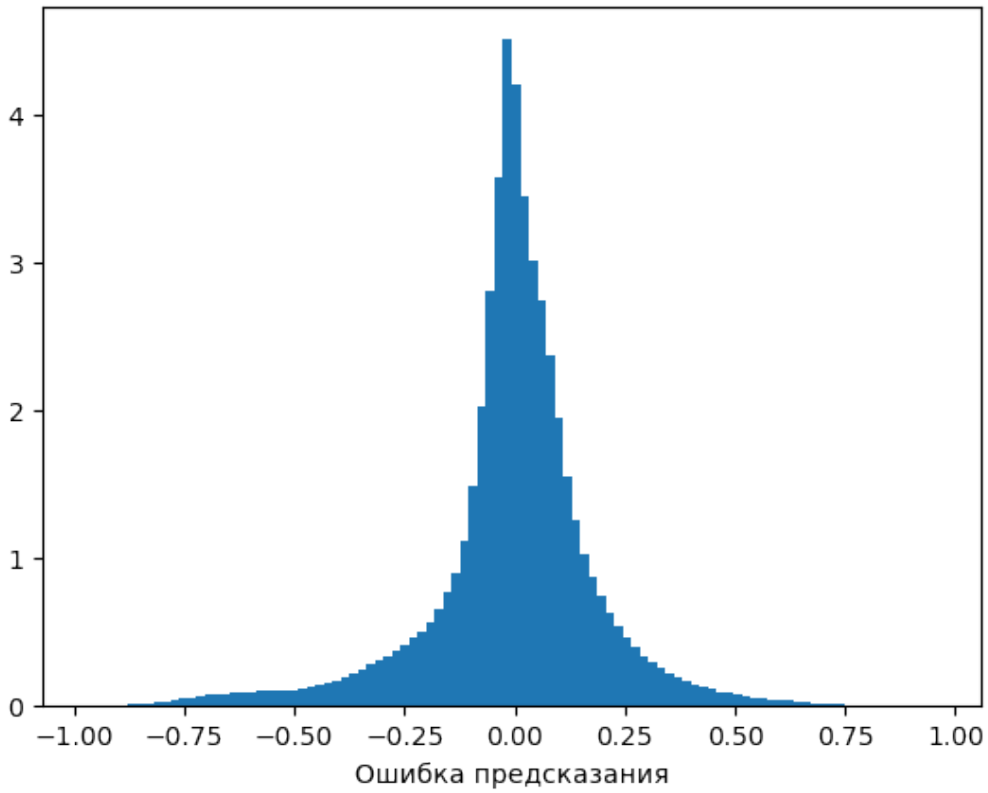

Рисунок 5.3 — Распределение ошибок предсказания нового потребления из отрезка [0.0001, 0.9999]

сказываемого распределения. При обучении минимизируется расстояние Кульбака — Лейблера от эмпирического распределения до предсказываемого, или логарифм функции правдоподобия истинного потребления процессорного времени в предсказанном распределении. В силу описанных в предыдущем абзаце особенностей распределения потребления, было выбрано следующее параметрическое семейство вероятностных распределений:

$$p(x|\pi_0, \pi_x, \pi_1, \mu, \alpha) = \pi_x \cdot \frac{\alpha}{2Z(\mu, \alpha)} \exp\left(-\alpha|x - \mu|\right) + \pi_0 \cdot \delta(x = 0) + \pi_1 \cdot \delta(x = 1)$$

$$Z(\mu, \alpha) = 1 - \frac{1}{2}\exp\left(-\alpha(1 - \Delta - \mu)\right) - \frac{1}{2}\exp\left(\alpha(\Delta - \mu)\right)$$

Дельта-функция Дирака делает вычисление правдоподобия затруднительным, поэтому дельта-функции на концах единичного отрезка были заменены узкими ($\Delta = 10^{-4}$) равномерными распределениями. $Z(\mu, \alpha)$ нормирует распределение Лапласа, чтобы учесть ограниченность носителя.

5.4.1 Особенности обучения

Для успешного обучения построенной модели оказалось важным применение ряда приёмов.

Ключевым моментом в обучении было строгое разделение тренировочных примеров, применённых для обучения предсказания параметра α, и примеров, применённых для обучения остальных выходов нейронной сети. Итерации этих двух процессов обучения проводились на одной сети попеременно. Для реализации такой особенности обучения, обратное распределение градиента через выходы сети условным образом прерывалось. Включая распространение градиента через часть предсказываемых параметров распределения, можно добиться уточнения оценки именно этих выходов. Такая техника необходима, так как на тренировных примерах ошибки оказываются заниженными, и для оценки степени неопределённости на новых данных необходимо использовать отдельные обучающие примеры.

В этой сети также были использованы все приёмы, описанные в разделе 2.3. Особенно большой прирост к точности дало использование dropout-слоёв.

Коэффициент детерминации обученной модели, если использовать предсказание средним от полученного распределения, оказался равным 0.4522. Для сравнения, нейронная сеть, напрямую обученная для минимизации среднеквадратичной ошибки, показала $R^2 = 0.4554$.

5.5 Прототип планировщика

Заключительным этапом работы является проверка применимости предсказываемых вероятностных распределений для распределения нагрузки. В главе 4 про симуляцию описана и обоснована симуляционная среда. Её устройство таково, что планировщик имеет максимально ограниченный набор возможных действий. Всё, что может делать планиров-

щик - это бинарные решения о запуске очередной нагрузки на единственной симулируемой физической машине. Возможности выбрать нагрузку или физическую машину для запуска у планировщика нет. В действительности, у планировщика будет больше свободы, и, соответственно, больше потенциала для оптимизации.

5.5.1 Базовый планировщик

В качестве базового планировщика был взят планировщик, который запускает очередную нагрузку только в момент, когда после её добавления суммарная квота не будет превышать ресурсы сервера. Эта политика консервативна, при ней заведомо невозможна перегрузка. Доля использованных ресурсов будет сравниваться с долей использования процессорного времени этой базовой политикой.

5.5.2 Предложенный планировщик

Разрабатываемый планировщик располагает оценками распределения потребления процессора в разные интервалы времени в будущем, 4 вероятностных распределения для каждой нагрузки. На основании этой информации, и описания нового кандидата на исполнение, необходимо принять решение о запуске или откладывании запуска в данный момент.

Планировщик принимает свое решение на основании оценки вероятности перегрузки в любой из четырёх горизонтов прогнозирования, порог вероятности является единственным параметром алгоритма, требующим настройки. Потребление нагрузок, для которых ещё не набралось ни единой записи в истории, предсказывается верхней границей квоты. Так происходит и с нагрузкой, которая является кандидатом на запуск. Требуется оценить вероятность превышения заданного значения суммой случайных величин известных распределений.

Одним из рассмотренных и отвергнутых подходов было использование череды свёрток гистограмм распределений. Используется вектор вероятностей попадания текущей суммы случайных величин в ячейки достаточно частой сетки, для прибавления очередной нагрузки производится свёртка её распределения и распределения текущей суммы. Для работы этого подхода сетка должна быть настолько частой, чтобы было возможно достаточно детально приблизить распределения потребления нагрузок даже с малыми квотами. Число одновременно работающих нагрузок измеряется десятками, и может достигать 50-60. В этих условиях, вычислительные затраты на работу такого варианта планировщика оказались неприемлемыми.

Одновременно получить высокую скорость выполнения и приемлемую точность аппроксимации удалось при помощи метода Монте-Карло. В ходе работы, из каждого распределения генерирует заданное количество реализаций (использовалось 10000), эти реализации суммируются для всех нагрузок, и вычисляется доля превышающих порог сумм. В таком виде, для получения оценки порядка 10^{-6} необходимо использовать от 10^6 реализаций, что снова оказывается слишком затратным. Однако, при использовании выборки по значимости (стр. 14), это ограничение снимается.

Опишем алгоритм оценки вероятности перегрузки для отдельного горизонта предсказания поподробнее. Вначале алгоритму доступны параметры распределения будущих потреблений процессорного времени, эти распределения имеют различные носители (согласно квотам). Обозначим плотность распределения для нагрузки i как f_i, порог суммарного потребления обозначим через t. В этих обозначениях, искомая вероятность записывается как:

$$\gamma = \int_{c_1=0}^{\mathrm{quota}_1} \cdots \int_{c_N=0}^{\mathrm{quota}_N} f_1(c_1) \cdots f_N(c_N) \left[\sum_{i=1}^{N} c_i > t \right] \mathrm{d}c_1 \cdots \mathrm{d}c_N$$

где

$$[expr] = \begin{cases} 1, & \text{if } expr \\ 0, & \text{else} \end{cases}$$

Использование выборки по значимости предполагает модификацию распределений. Для получения реализаций будущего потребления процессорного времени отдельными нагрузками, было использовано распределение с плотностью:

$$\tilde{f}_i(c_i) = \frac{1}{Z_i} f_i(c_i) \cdot \left(1 + \frac{c_i}{\text{quota}_i}\right)^2$$

Переходя к преобразованным распределениям, искомую вероятность γ можно эквивалентно переписать как:

$$\gamma = \int_{c_1=0}^{\text{quota}_1} \cdots \int_{c_N=0}^{\text{quota}_N} \tilde{f}_1(c_1) \cdots \tilde{f}_N(c_N) \frac{\left[\sum_{i=1}^N c_i > t\right]}{\prod_{i=1}^N \left(1 + \frac{c_i}{\text{quota}_i}\right)^2 Z_i^{-1}} \mathrm{d}c_1 \cdots \mathrm{d}c_N$$

Для вычисления нормировочных констант Z_i преобразованных распределений использовалось суммирование по сетке из 10000 точек на отрезках $[0, \text{quota}_i]$.

Хотя интеграл аналитически и остался в точности таким же, при его оценке методом Монте-Карло большие значения c_i имеют увеличенную вероятность реализации, а значит индикаторная функция в числителе будет ненулевой для существенной доли примеров, даже если изначально величина $\frac{1}{P[\sum_{i=1}^N c_i > t]}$ была много меньше количества используемых реализаций.

Итоговый алгоритм оценки γ состоит из генерации $\{c_i^j \sim \tilde{f}_i\}_{j=1..M, i=1..N}$ (M = 10000), и вычисления:

$$\gamma \approx \frac{1}{M} \sum_{j=1}^{M} \frac{\left[\sum_{i=1}^N c_i^j > t\right]}{\prod_{i=1}^N \left(1 + \frac{c_i^j}{\text{quota}_i}\right)^2 Z_i^{-1}}$$

5.6 Результаты симуляции

Симулировался сервер, имеющий максимальную в кластере производительность процессора. Нагрузки подавались на исполнение в зафиксированном случайном порядке, они брались из тестовой части выборки, уже после завершения всех нагрузок, использованных при обучении модели.

На рисунке 5.4 показано сравнение суммарного потребления процессорного времени при управлении базовым планировщиком, и предложенным планировщиком с порогом вероятности перегрузки 10^{-3}. На графике 5.5 представлено изменение количества единовременно исполняемых нагрузок для тех же выполнений симуляции.

Сравнение хода симуляции для всех рассмотренных планировщиков приведено на графиках 5.6 и 5.7. На них изображены бегущее среднее суммарного потребления процессорного времени и количество одновременно выполняемых нагрузок.

В таблице 5.1 приведено сравнение показателей работы планировщиков. Видно, что доля моментов времени с перегруженным сервером следует за выставляемым порогом, и по порядку величины совпадает с ним. Это подтверждает высокую информативность параметра разработанного прототипа планировщика.

Планировшик	Среднее использование ЦПУ	Среднее превышение доступного ЦПУ	Доля моментов времени с превышением доступного ЦПУ
Базовый	0.529	0	0
Прототип, p=10^{-3}	0.757	$4.87 \cdot 10^{-5}$	$1.65 \cdot 10^{-3}$
Прототип, p=10^{-4}	0.718	$3.34 \cdot 10^{-6}$	$4.13 \cdot 10^{-4}$
Прототип, p=10^{-5}	0.687	$7.35 \cdot 10^{-8}$	$6.89 \cdot 10^{-5}$

Таблица 5.1

Сравнение результатов симуляции

Таким образом, использование предложенного планировщика позволило увеличить использование ресурса процессорного времени на 29.8-43.0 %, явно контролируя риск нехватки ресурсов.

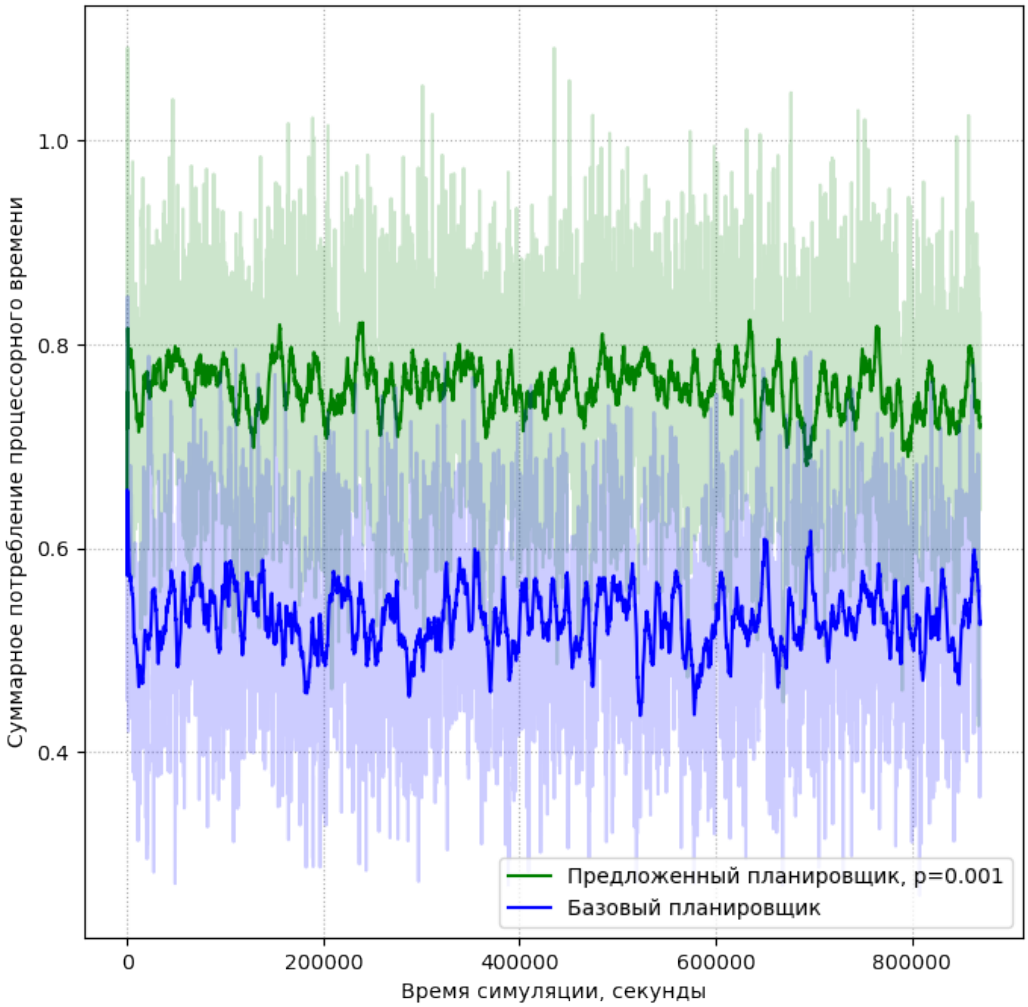

Рисунок 5.4 — Сравнение суммарного потребления процессорного времени

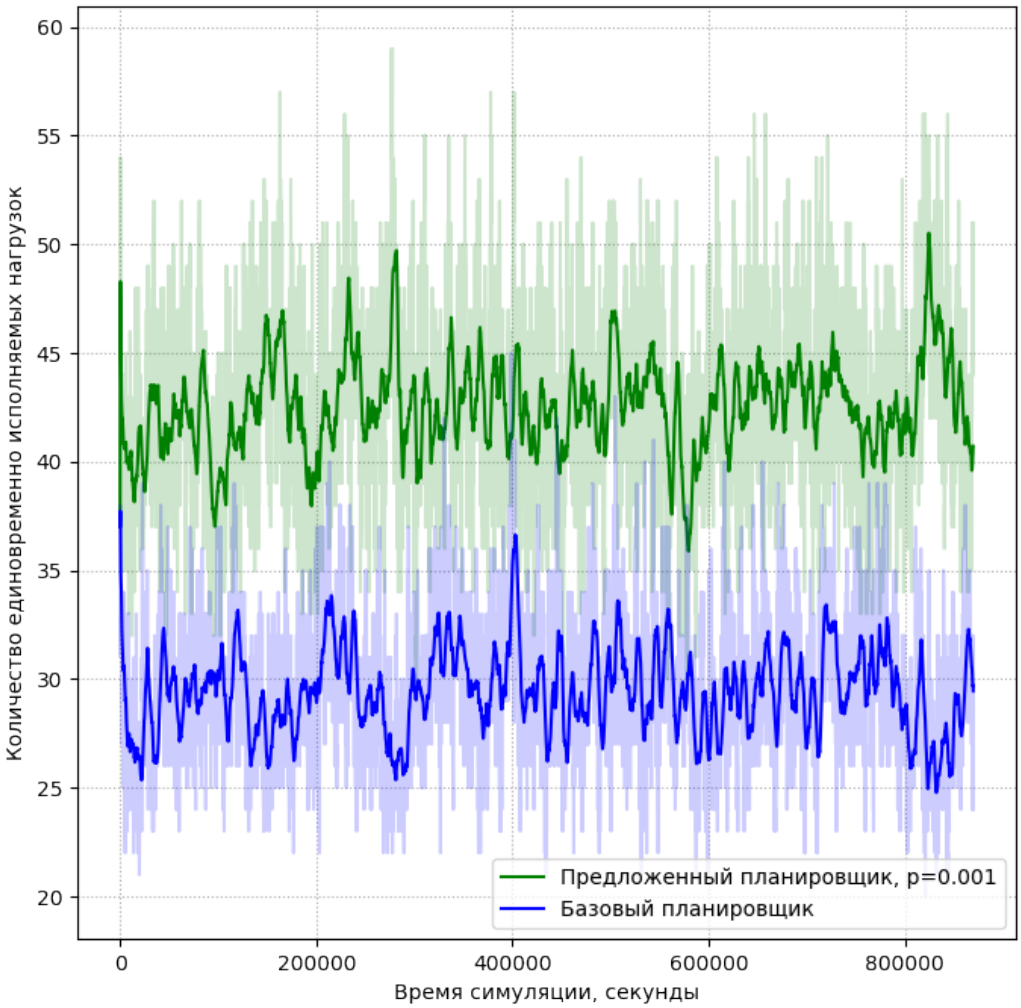

Рисунок 5.5 — Сравнение количества выполняемых нагрузок

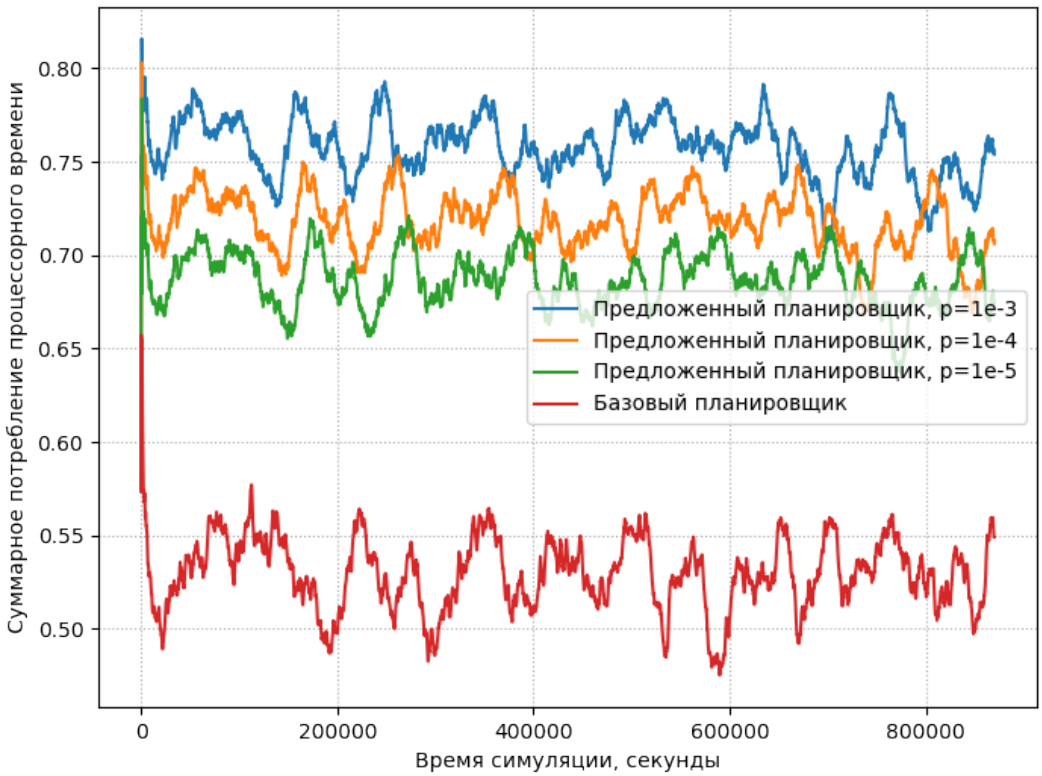

Рисунок 5.6 — Сравнение суммарного потребления процессорного времени

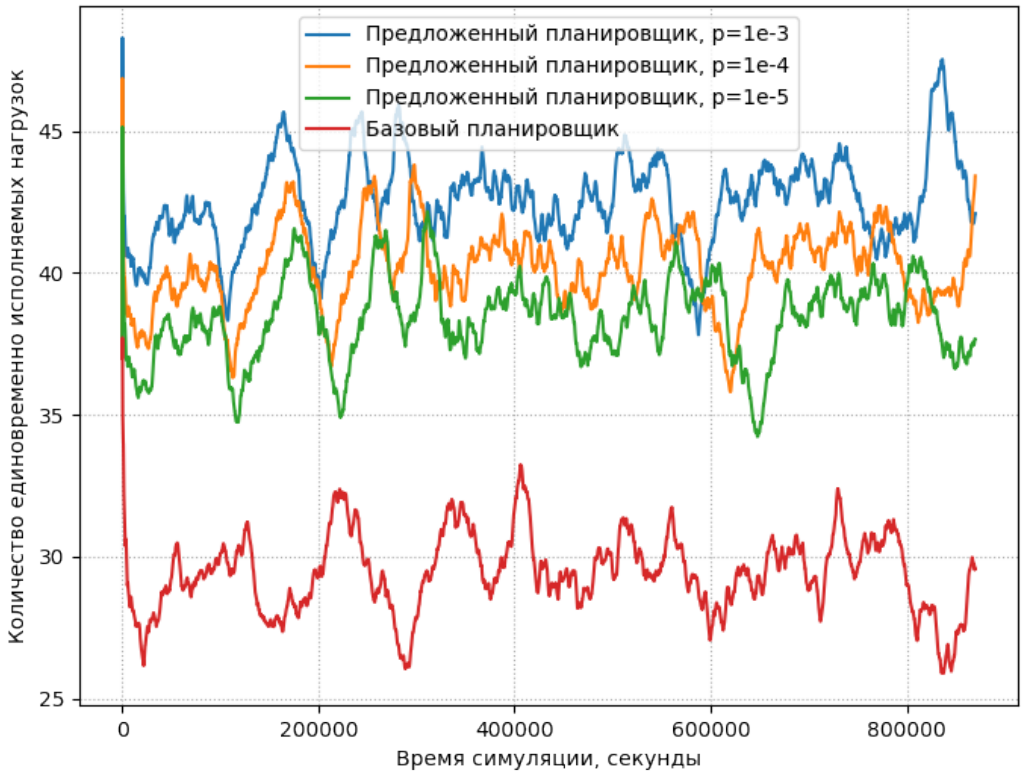

Рисунок 5.7 — Сравнение количества выполняемых нагрузок

Глава 6. Обсуждение результатов

В ходе работы был получен ряд результатов.

1. Выполнен обзор применяемых методов предсказания потребления процессорного времени в виртуальных средах, как для отдельных нагрузок, так и для совокупного потребления ресурсов сервером. В большинстве случаев предпочтение отдавалось линейным моделям и/или моделям, использующим лишь одномерный ряд истории потребления процессорного времени.
2. Обработаны и структурированы для дальнейшей работы публично доступные данные месячной активности кластера из 12.5 тыс. серверов компании Google. Произведён выбор полезных для предсказания признаков.
3. Оценено качество работы алгоритмов в рассматриваемом приложении, предложены их модификации, проведено сравнение. Нелинейные модели и многофакторные модели, учитывающие различные показатели работы нагрузки, оказались существенно точнее альтернатив.
4. Реализована симуляционная среда с прототипом алгоритма планирования нагрузки, оценивающая применимость вероятностных моделей предсказания процессорной нагрузки на исторических данных.
5. Предложен алгоритм расчёта распределения вероятностей будущего потребления процессорного времени. Разработан прототип планировщика запуска нагрузок, использующий предсказанные распределения. По результатам тестирования алгоритмов прирост среднего использования процессорного времени хостом оказался в диапозоне 29-42% на исторических тестовых данных, причём без существенной потери качества обслуживания. Алгоритм планирования позволяет напрямую контролировать вероятность нехватки процессорного времени.

Результаты, перечисленные в пункте 5 являются ключевыми для этой работы.

Отметим всю условность полученного прироста производительности. С одной стороны, планировщик был крайне ограничен в спектре своих возможных действий. Имея свободу сопоставлять пары сервер – нагрузка для множества серверов и множества нагрузок, заведомо возможно получить результаты не хуже, чем когда за планировщика это сопоставление уже сделано волей случая. С другой стороны, в ходе симуляции был сделан ряд упрощений. Главное упрощение состоит в рассмотрении исключительно потребления процессорного времени. В реальности, при балансировании нагрузки придётся учитывать также потребление оперативной памяти, операции с диском и использование сетевого канала. Детальное симулирование работы с оперативной памятью представляется куда более сложной задачей, чем отдельное рассмотрение процессорного времени. Другим упрощением является игнорирование возможной взаимной зависимости между нагрузками. Наличие корреляций между потреблениями процессорного времени нагрузками на одной физической машине потенциально может увеличить вероятность нехватки процессорного времени, как следствие нарушения предположения о независимости. С другой стороны, учёт связи нагрузок может позволить учесть корреляцию в вероятностной модели, и увеличить точность предсказания на счёт дополнительной информации от других нагрузок.

Однако, целью работы была демострация возможности эффективно использовать предсказываемое вероятностное распределение для размещения нагрузки. Точности модели предсказания хватило, чтобы на его основании планировщик мог добиватся желаемой вероятности перегрузки. В результате планировщик предоставлял желаемый контроль рисков, что подтверждает принципиальную применимость предложенной конструкции.

Заключение

Для максимальной точности предсказания потребления процессорного времени линейных моделей недостаточно, искусственные нейронные сети показывают значительно более высокую точность. В число независимых переменных крайне важно включить история потребления процессорного времени, но дополнительные параметры также дают некоторый прирост точности.

Для обезопашенного от нехватки ресурсов распределения нагрузки, необходима не точечная оценка будущего потребления, а оценка всего его вероятностного распределения. Наличие такой оценки для каждой нагрузки позволяет достаточно точно оценивать вероятность нехватки рассматриваемого ресурса, что экспериментально подтвердилось в ходе работы. Для предсказания распределения можно использовать вариационные нейронные сети.

В качетсве продолжения данной работы, разумно включить в рассмотрение другие системные ресурсы. Такое расширение, вероятно, потребует наличие более детальных исторических данных, либо доступ реально исполняемым нагрузкам. Конечной целью является эффективная балансировка реальных нагрузок.

Использование более частых измерений может улучшить точность предсказания и уменьшить задержки в реагировании планировщика. Учёт взаимодействия между нагрузками также может увеличить точность. И наконец, точность предсказания может быть возможно повычить с использованием рекуррентных нейронных сетей (хранящих внутреннее состояние).

Список литературы

1. *Dhiman Gaurav, Mihic Kresimir, Rosing Tajana.* A system for online power prediction in virtualized environments using gaussian mixture models // Design Automation Conference (DAC), 2010 47th ACM/IEEE / IEEE. — 2010. — Pp. 807–812.

2. *Meisner David, Gold Brian T, Wenisch Thomas F.* PowerNap: eliminating server idle power // ACM Sigplan Notices / ACM. — Vol. 44. — 2009. — Pp. 205–216.

3. *Pakbaznia Ehsan, Pedram Massoud.* Minimizing data center cooling and server power costs // Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design / ACM. — 2009. — Pp. 145–150.

4. *Di Sheng, Kondo Derrick, Cirne Walfredo.* Characterization and Comparison of Google Cloud Load versus Grids. http://hal. archives-ouvertes. fr/hal-00705858. — 2012.

5. *Di Sheng, Kondo Derrick, Cirne Walfredo.* Host load prediction in a Google compute cloud with a Bayesian model // Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis / IEEE Computer Society Press. — 2012. — P. 21.

6. *Kalyvianaki Evangelia, Charalambous Themistoklis, Hand Steven.* Adaptive resource provisioning for virtualized servers using Kalman filters // *ACM Transactions on Autonomous and Adaptive Systems (TAAS).* — 2014. — Vol. 9, no. 2. — P. 10.

7. *Kalyvianaki Evangelia, Charalambous Themistoklis, Hand Steven.* Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters // Proceedings of the 6th international conference on Autonomic computing / ACM. — 2009. — Pp. 117–126.

8. Adaptive control of virtualized resources in utility computing environments / Pradeep Padala, Kang G Shin, Xiaoyun Zhu et al. // ACM

SIGOPS Operating Systems Review / ACM. — Vol. 41. — 2007. — Pp. 289–302.

9. *Berger James O.* Statistical decision theory and Bayesian analysis. — Springer Science & Business Media, 2013.

10. *Di Sheng, Kondo Derrick, Cirne Walfredo.* Google hostload prediction based on Bayesian model with optimized feature combination // *Journal of Parallel and Distributed Computing.* — 2014. — Vol. 74, no. 1. — Pp. 1820–1832.

11. *Dinda Peter A, O'Hallaron David R.* An evaluation of linear models for host load prediction // High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on / IEEE. — 1999. — Pp. 87–96.

12. *Dinda Peter A.* The statistical properties of host load // International Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers / Springer. — 1998. — Pp. 319–334.

13. *Dinda Peter A, O'hallaron David R.* Host load prediction using linear models // *Cluster computing.* — 2000. — Vol. 3, no. 4. — Pp. 265–280.

14. *Domingos Pedro, Pazzani Michael.* On the optimality of the simple Bayesian classifier under zero-one loss // *Machine learning.* — 1997. — Vol. 29, no. 2. — Pp. 103–130.

15. *Dunham Margaret H.* Data mining: Introductory and advanced topics. — Pearson Education India, 2006.

16. *Farahat MA, Talaat M.* Short-Term Load Forecasting Using Curve Fitting Prediction Optimized by Genetic Algorithms // *International Journal of Energy Engineering.* — 2012. — Vol. 2, no. 2. — Pp. 23–28.

17. *Kalyvianaki Evangelia, Hand Steven.* Applying Kalman filters to dynamic resource provisioning of virtualized server applications // Proc. 3rd Int. Workshop Feedback Control Implementation and Design in Computing Systems and Networks (FeBid). — 2008. — P. 6.

18. Adaptive workload prediction of grid performance in confidence windows / Yongwei Wu, Kai Hwang, Yulai Yuan, Weiming Zheng // *IEEE Transactions on Parallel and Distributed Systems*. — 2010. — Vol. 21, no. 7. — Pp. 925–938.

19. Load prediction using hybrid model for computational grid / Yongwei Wu, Yulai Yuan, Guangwen Yang, Weimin Zheng // Grid Computing, 2007 8th IEEE/ACM International Conference on / IEEE. — 2007. — Pp. 235–242.

20. *Yoas Daniel W, Simco Greg*. Resource utilization prediction: A proposal for information technology research // Proceedings of the 1st Annual conference on Research in information technology / ACM. — 2012. — Pp. 25–30.

21. *Zhang Yuanyuan, Wei SUN, Inoguchi Yasushi*. CPU load predictions on the computational grid // *IEICE TRANSACTIONS on Information and Systems*. — 2007. — Vol. 90, no. 1. — Pp. 40–47.

22. *Bellosa Frank*. The benefits of event: driven energy accounting in power-sensitive systems // Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system / ACM. — 2000. — Pp. 37–42.

23. *Contreras Gilberto, Martonosi Margaret*. Power prediction for Intel XScale/spl reg/processors using performance monitoring unit events // Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on / IEEE. — 2005. — Pp. 221–226.

24. *Li Tao, John Lizy Kurian*. Run-time modeling and estimation of operating system power consumption // ACM SIGMETRICS Performance Evaluation Review / ACM. — Vol. 31. — 2003. — Pp. 160–171.

25. ClusterData2011_2 traces. — https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md. — Accessed: 22.05.2017.

26. *Hornik Kurt*. Approximation capabilities of multilayer feedforward networks // *Neural networks*. — 1991. — Vol. 4, no. 2. — Pp. 251–257.

27. *Murphy Kevin P.* Machine learning: a probabilistic perspective. — MIT press, 2012.

28. Dropout: A simple way to prevent neural networks from overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky et al. // *The Journal of Machine Learning Research.* — 2014. — Vol. 15, no. 1. — Pp. 1929–1958.

29. *Baldi Pierre, Sadowski Peter J.* Understanding dropout // Advances in Neural Information Processing Systems. — 2013. — Pp. 2814–2822.

30. *Ioffe Sergey, Szegedy Christian.* Batch normalization: Accelerating deep network training by reducing internal covariate shift // *arXiv preprint arXiv:1502.03167.* — 2015.

31. *Kudinova Marina, Melekhova Anna, Verinov Alexander.* CPU utilization prediction methods overview // Proceedings of the 11th Central & Eastern European Software Engineering Conference in Russia / ACM. — 2015. — P. 7.

32. *Bobroff Norman, Kochut Andrzej, Beaty Kirk.* Dynamic placement of virtual machines for managing sla violations // Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on / IEEE. — 2007. — Pp. 119–128.

33. Profiling and modeling resource usage of virtualized applications / Timothy Wood, Ludmila Cherkasova, Kivanc Ozonat, Prashant Shenoy // Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware / Springer-Verlag New York, Inc. — 2008. — Pp. 366–387.

34. Empirical prediction models for adaptive resource provisioning in the cloud / Sadeka Islam, Jacky Keung, Kevin Lee, Anna Liu // *Future Generation Computer Systems.* — 2012. — Vol. 28, no. 1. — Pp. 155–162.

35. Omega: flexible, scalable schedulers for large compute clusters / Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek,

John Wilkes // Proceedings of the 8th ACM European Conference on Computer Systems / ACM. — 2013. — Pp. 351–364.

36. Large-scale cluster management at Google with Borg / Abhishek Verma, Luis Pedrosa, Madhukar Korupolu et al. // Proceedings of the Tenth European Conference on Computer Systems / ACM. — 2015. — P. 18.

37. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control / Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, Erik Elmroth // Proceedings of the 3rd workshop on Scientific Cloud Computing Date / ACM. — 2012. — Pp. 31–40.