Найбільш очікуваний результат – це негайне завершення в найкоротші терміни. Завдяки цьому процесу всі ці діагностичні параметри можуть бути виміряні в єдиному стані. Отже, необхідність використання системи контролю та управління визначається вирішенням вищезазначених та інших проблем, що виникають при роботі батареї.

Список використаних джерел:

1. Багоцкий В.С. Химические источники тока / В.С. Багоцкий, А.М. Скундин. – М.: Энергоиздат, 2007. – 360 с.

2. Ефимов О.Н. Новые материалы для литиевых аккумуляторов / О.Н. Ефимов, Д.Г. Белов, Г.П. Белов и др. / Машиностроитель. – 2004. – С. 24–28.

3. Потупчик С. Литий-полимерные (Li-Pol) аккумуляторы [Електронний ресурс] / С. Потупчик / RCDesign. – 2009. – Режим доступу: http://www.rcdesign.ru/articles/engines/lipol/

4. Скундин А.М. Современное состояние и перспективы развития литиевых аккумуляторов / А.М. Скундин, О.Н. Ефимов. – 2005. – 420 с.

5. Ярмоленко О.В. / Успехи химии. – 2002. – С. 378–398.

6. Вайлов, А.М. Автоматизация контроля и обслуживания аккумуляторных батарей / А.М. Вайлов, Ф.И. Эйгель. – М.: Связь, 2006. – 156 с.

Ткаченко Е.А., Андрущенко Н.А.
студенты,
Харьковский национальный университет радиоэлектроники

GOLANG – НОВЫЙ ВЗГЛЯД НА ПАРАЛЛЕЛИЗМ

Высокий интерес к распределенным и многопроцессорным системам, а также необходимость в обработке большого количества данных, потребовало от разработчиков поиска новых решений в области параллельного программирования для оптимизации вычислительных ресурсов. Компания Google представила новый продукт – язык Golang, который нацелен стать новым универсальным языком программирования, дополнив C++ такими возможностями как: автоматическое управление памятью, безопасность типов и новый подход к параллельным вычислениям [1]. Однако действительно ли

новая модель параллелизма позволит повысить продуктивность программ и составит конкуренцию языкам широкого применения?

Традиционный подход к параллелизму, как в C++, заключается в конструировании объекта std::thread и использовании средств управления потоками [2]. Проблемы разделения данных между потоками решаются регулирующими правилами общения. Такой подход к управлению потоков может быть громоздким. Go поощряет другой подход, в котором общие значения передаются по каналам и, фактически, никогда активно не разделяются отдельными потоками исполнения. Только одна программа имеет доступ к значению в любой момент времени. Гонки данных не могут происходить, по замыслу. Для поощрения такого мышления существует лозунг: «Не общайтесь, разделяя память. Вместо этого делитесь памятью, общаясь» [3].

Также, Go использует различную от потоков модель общей памяти – *Go-процедуры.* Go-процедура во многом похожа на поток, но имеет более простую модель: это функция, выполняющаяся одновременно с другими Go-процедурами в том же адресном пространстве. Она легковесная, стоящая чуть больше, чем выделение стекового пространства. Go-процедуры мультиплексируются в несколько потоков ОС, так что, если одна из них нуждается в блокировке, например во время ожидания ввода-вывода, другие продолжают работать. Их дизайн скрывает многие сложности создания потоков и управления ими [4].

В отличии от примитивов синхронизации, используемых в C++: семафоров, блокировок и условных переменных, Go решает проблему синхронизации через использование встроенных примитивов под названием «канал». Каналы – это типизированные «трубы», по которым можно посылать и получать значения. По умолчанию отправление и получение блокируются до тех пор, пока другая сторона не готова. Это позволят Go-процедурам синхронизироваться без явного использования блокировок и условных переменных.

Для сравнения традиционного и Go подхода к параллельному программированию был проведен расчет числа π по методу Монте-Карло. Суть метода Монте-Карло сводится к наипростейшему перебору точек на плоскости [5]. Идея расчёта состоит в том, чтобы взять квадрат со стороной $a = 2R$, вписать в него круг радиусом R и наугад ставить точки внутри квадрата. Вероятность $P1$ того, что точка попадет в круг

равна отношению площади круга и квадрата: $P1 = \frac{S\text{круг.}}{S\text{квадр.}} = \frac{\pi R^2}{a^2} = \frac{\pi R^2}{(2R)^2} = \frac{\pi}{4}$ (рис. 1).

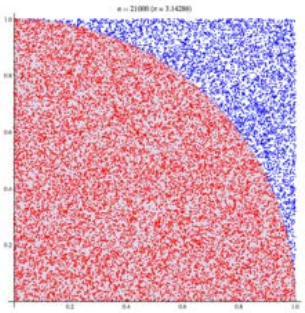

Рис. 1. Вычисления числа π методом Монте-Карло

В ходе эксперимента был реализован алгоритм (рис. 2) с помощью языков Go и C++.

```
getPI_MonteCarlo(number){
    x,y;
    count = 0;
    for(i = 0; i < number; i++){
        x = rand();
        y = rand();
        if(x*x + y*y <= 1.0){
            count++;
        }
     }
    return(4 * count) / number;
}
```

Рис. 2. Программный алгоритм вычисления числа π методом Монте-Карло

В параллельной программе подсчета числа π на языке Go использовалась функция runtime.GOMAXPROCS(*nCPU), которая устанавливает количество потоков, а также был определен канал для записи частичных результатов, полученных в параллельных потоках. В

программе на языке C++ был использован OpenMP, для создания параллельной программы.

Результаты измерений времени выполнения программ были занесены в таблицу 1. Как мы видим, параллельная программа на Go выполнилась в 1,7882 раза быстрее, чем на C++. То есть, преимущество в скорости достается Go.

Таблица 1

Сравнение времени выполнения программ

Язык	Время, с	Количество повторов в цикле, раз
Go	9,3838	1 000 000 000
C++	16,7808	1 000 000 000

Подводя итог, Go – новый язык, который предоставляет возможности, упрощающие многие аспекты программирования, особенно в осуществлении параллельных программ, и повышающие их производительность. Простая модель общей памяти, а также использование каналов для межпроцессорного взаимодействия, увеличивает сложность выполняемых задач и способствует росту производительности, что обеспечивает конкурентоспособность языка Go.

Список использованных источников:

1. Алан А.А. Донован, Брайан У. Керниган Язык программирования Go**.** 2019. – ISBN 978-5-907114-21-0. – 432 с.

2. Уильямс Э. *Параллельное программирование на C++ в действии. Практика разработки многопоточных программ //* Компьютеры и Интернет**.** – 2012. – ISBN 0-14-100051-1. – 672 с.

3. Go vs. C: A Language comparison of Concurrent Programming Features. –2014. – URL: http://dead10ck.github.io/2014/12/15/go-vs-c.html

4. Effective Go. URL: https://golang.org/doc/effective_go.html#parallel

5. Швец А.Н. Perl. Примеры программ. 2019. URL: http://mech.math.msu.su/~shvetz/54/inf/perl-problems/chPi_sIdeas.xhtml