修士学位論文要約（令和４年３月）

再エネデータセンタのジョブスケジューリングに関する研究

加藤　大裕
指導教員：山田　博仁

A Study on Job Scheduling for Renewable Energy Data Centers

Daisuke KATO
Supervisor: Hirohito YAMADA

In recent years, power consumption in data centers has been increasing in line with the growth of data traffic, and this trend is expected to continue, with power consumption of 500,000 TWh expected to be consumed by 2050 for data-related purposes only. Therefore, from the viewpoint of energy conservation, distributed data centers using container-type data centers, rather than large data centers, are effective.

In order to increase the reenergy usage rate of distributed data centers, when the recharge rate of the storage batteries of containerized servers that are part of a distributed data center connected to a DC microgrid drops, the data center jobs are moved to reduce power consumption and distribute the load to servers that can afford to be more power-efficient. In this study, we report on the construction of the foundation of the above system and the verification of its effectiveness using a server with a one-board PC.

1.　はじめに

通信情報の増加に従い，データセンターの消費電力が増加している．また，最新の情報機器を用いた場合，2050 年に情報関連のみで年間 500,000TWh の消費電力が必要と予測される．また，新しいテクノロジーの普及によって数年でエネルギー需要が 2 倍に膨れ上がりエネルギー利用の均衡を保てなくなる可能性[1][2]が存在している．そのため，データセンターの再生可能エネルギーの使用率を上げることで，データセンターの省エネルギー化を勧める必要がある．データセンターの蓄電池の充電率が低下した際，マイクログリッド間で電力融通をする事により，蓄電池の充電を進める事ができる．しかし，この手段では，電力融通の際に電力ロスが発生する他，電力的に接続されているグリッド間でのみ可能な方法である．一方で，ジョブを他のデータセンターに再分配し，情報処理負荷を減らす手段がある．この手法は再分配のシステムを作る必要があるもののデータのやり取りのみであるため，電力的なロスが存在しない他，地理的に離れた土地であっても処理を肩代わりしてもらうことが可能である．そのため，本研究では蓄電池の充電率や PV セルの発電量・発電予測などをもとにデータセンター内のジョブを動的に分配・再分配するシステムの基礎の構築を行なったので報告する．

2.　使用機器およびジョブスケジューリングの概要

2. 1.　本研究における使用機器の概要

本研究では，ジョブスケジューリングによる電力消費の軽減が可能か調査するために CPU 使用率と消費電力の関係の計測を計測する必要があった．そのため，コンテナサーバーやサーバーユニットではネットワークカードや記憶デバイスなどが存在しているため，CPU 使用率と消費電力の関係を計測するには不向きであると判断したことに加え，蓄電池でサーバーを稼働させる必要があったため，Raspberry Pi 4 を用いた実験を行った．

2. 2.　ジョブスケジューリングの概要と提案手法について

サーバの動作環境に左右されることなく動作させるためにコンテナアプリケーションとして Docker[3]とコンテナオーケストレーションソフトとして Kubernetes[4]を用いた環境を構築した．また，本環境はモバイルバッテリーで駆動するコンテナクラスタになっている．Kubernetes にはジョブスケジューリングを行う機能として kube-scheduler が備わっているが，その基本機能だけでは電源状況に基づくジョブスケジューリングを実現できないので，scheduler framework[5]を用いてプラグインを作成し，プラグインを加えたカスタム scheduler を作成する必要がある．ジョブスケジューリングの方法としては，充電率(SoC)の最も高いサーバ(Node)に Pod を分配していくプラグインを作成し，そのプラグインを加えたカスタム Scheduler を用いて実験を行った．カスタム Scheduler のフローチャートを図 1 に示す．また，赤い点線で囲まれた部分が作成したプラグインによる処理である．また，Kubernetes に

はジョブの再配分を行う機能が備わっていないため，descheduler[6]というKubernetesのアプリケーションで補完することにより，目指す機能を実現した．

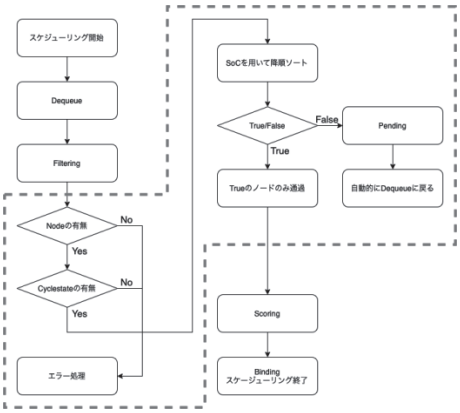

図 1　ジョブスケジューリングのフローチャート

3.　CPU 使用率と消費電力の関係について

パソコンやサーバマシンの電力消費には，情報処理負荷に応じて増減する変動分と，負荷の大小に関わらず一定の電力を消費する固定分とがある．図 1 は，各種マシンの CPU 使用率と消費電力との関係を測定したもの(ただし，機種ごとの最大電力で規格化している)であるが，変動分は大きいものでは65%にもなることが分かる．

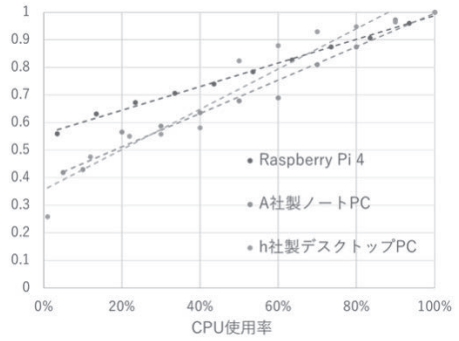

図 2　最大電力で規格化した CPU 使用率と消費電力との関係

4. 提案手法を用いたジョブスケジューリングの検証

構築したシステムを用いて，まずは 2 台の Raspberry Pi 間でのジョブスケジューリング動作を検証した．両 Raspberry Pi の電源に容量 20Ah のモバイルバッテリーを使用し，SoC を各々100%と70%にした上で，どちらか一方の充電率が 40%に低下した時点で，ジョブスケジューリングを開始し，充電率の高い方の Node へ Pod が再配分される動作を確認できた．また，同様の実験を 3 台の Raspberry Pi 間でも行った．その結果，ジョブスケジューリングを行わなかった場合，モバイルバッテリーの残量がゼロとなってマシンが停止するまでの稼働時間が 429 分であったのに対し，2 台の Raspberry Pi でのジョブスケジューリングを行なった場合は稼働時間が 672 分，3 台の場合では 678 分となり，稼働時間が 3 時間程度延びたことを確認できた(図 3)．また，負荷分散先の数に関わらず，消費電力を抑えたサーバーの動作時間に大きな違いは確認できなかった．

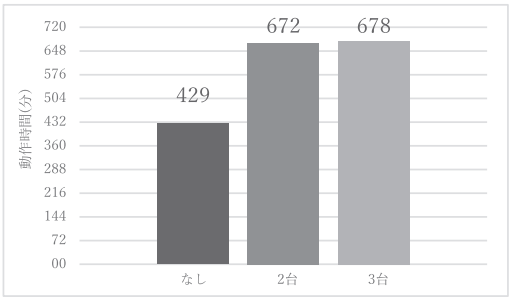

図 3　提案手法の有無によるサーバーの動作時間の違い

5.　まとめ

ワンボードマイコンによるコンテナ クラスタを構築し，Kubernetes によるジョブスケジューリングの機能を拡張するプラグインを作成，電源状況に応じたジョブスケジューリング動作を実現した．ジョブスケジューリングを行うことでマシンの稼働時間を大幅に伸ばすことに成功し，電源状況に応じたジョブスケジューリングの効果を確認することができた．

文献

1)　低炭素社会実現に向けた制作立案のための提案書，“情報化社会の進展がエネルギー消費に与える影響(Vol. 1)” -IT 機器の消費電力の現状と将来予測-“，科学技術振興機構低炭素社会戦略センター，2019 年 3 月

2)　低炭素社会実現に向けた制作立案のための提案書，“情報化社会の進展がエネルギー消費に与える影響(Vol. 2)” -IT 機器の消費電力の現状と将来予測-“，科学技術振興機構低炭素社会戦略センター，2021 年 2 月

3)　Docker，“Docker”，https://www. docker. com (アクセス日 2021 年 4 月 12 日)

4)　kubernetes，“kubernetes”，https://kubernetes. io (アクセス日 2021 年 4 月 13 日)

5)　kubernetes，“Scheduling Framework”，https://kubernetes. io/docs/concepts/scheduling-eviction/scheduling-framework/ (アクセス日 2021 年 4 月 30 日)

6)　kubernetes-sigs，“descheduler”，https://github. com/kubernetes-sigs/descheduler (アクセス日 2021 年 10 月 25 日)