リストアにおける進行状況にもとづくバックアップのCPU使用量の制限による時間の増加の抑制

石川裕人[1]　高橋風太[2]　串田 高幸[1]

概要：システムを運用するにあたって，データの損失や破損が発生した場合，システムの復旧を可能にするためにバックアップが行われる．また，バックアップからデータやシステムを復元するためにリストアが不可欠である．課題は，バックアップを行っている最中にリストアを行った際，バックアップを行っていないリストア単体の時間と比較してリストア時間が増加することである．提案手法では，ユーザはファイル転送に要する時間の 38 %を損失するとフラストレーションを感じることから，リストア単体で行った時間を基準値とし，基準値の 38 %増加した時間を許容時間として設定する．許容時間内にリストアを完了させるためにバックアップにおける CPU 使用量を仮想マシンのリストア時間によって制限する量を変更し，リストアを優先するソフトウェアを提案する．基礎実験では，研究室にある物理マシンを対象にリストア単体で行った時間が平均 163 秒であるのに対し，バックアップとリストアを同時に行った時間は平均 790 秒に増加した．この増加率は約 385 %となった．評価実験では，提案手法がバックアップを行っている最中にリストアを行った時間の増加を短縮するかどうかを確認する．

1. はじめに

背景

バックアップとは，データの損失や破損，誤操作，災害の予期しない事態に備えて，データのコピーを作成し，保存することである [1–3].

バックアップにはひとつの手法だけでなく種類があり，それぞれに長所と短所がある．主に使用されているバックアップ方法が 3 つあり，その方法はフルバックアップ，増分バックアップ，差分バックアップである [4, 5].

東京工科大学コンピュータサイエンス学部にある研究室の Cloud and Distributed Systems Laboratory(以下 CDSL) では，実際に 9 台の物理マシン (以下 PM) が運用されている．その中の 5 台を CDSL 所属の学生が自由に仮想マシン (以下 VM) をハイパーバイザ上に作成し作業を行う形で運用されている．CDSL では平日の 7:00 から 19:00 の間は研究室で作業を行うことができる．そして翌日の 3:00 にバックアップを行っている．バックアップを行う内容として，各 PM に入っている VM を毎日スナップショットを作成し，フルバックアップとして保存している．スナップショットとは，仮想環境において重要な状態の保持や，変更点の管理を行うための機能であり，例えばソフトウェアの更新やシステム変更の前後の状態で VM の完全なコピーを作成することである [6–8]．このコピーは，VM の状態を記録し，必要に応じてスナップショットを作成した過去に戻すことができる手段として利用される．スナップショットは，誤操作やシステムの変更前の安全な状態に戻すために使うことができる．また，ソフトウェアのテストや開発環境での利用にも柔軟に対応するために有用である [9].

データをバックアップする例として，北日本石油株式会社を挙げる．この企業の主な事業内容は石油製品の供給であるが，整備や洗車，中古自動車，部品の販売，損害保険の代理業といった業務もあるため，顧客情報の漏洩防止には細心の注意を払う必要がある．万が一の場合にも安全にデータ保持・復元ができるようにバックアップを導入した[*1].

バックアップはいつでもリストアできるようにすることが重要である [10]．バックアップするのには目的が存在する．例えば，システム感染，ハードウェアの問題，コンピュータの誤った電源オフ，ファイルの誤削除，ファイルシステムの損傷という問題から重要なデータを失わないようにし，アクセスが可能な状態を保つためである [11–13]．特に気を付けなければいけないのが，人為的ミスでデータ

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学院バイオ・情報メディア研究科コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404-1

[*1] https://www.backstore.jp/customers/kitanihon-oil.html

を損失，破損することである*2．このような理由から，リストアは決まった時間にすることが少ない．また，企業がバックアップを行う時間帯として業務時間外の深夜に行うことが基本である*3．

バックアップは大企業だけでなく，中小企業や大学でもストレージの管理として行っているため，どの組織においてもデータは重要であり，データがなければ業務を継続することは不可能である [14]．また，国際データコーポレーション (IDC) の報告によると，デジタルデータの量は 2006 年から 2010 年にかけて 50 %の割合で増加し，5 年間で 10 倍のストレージ領域が必要になった [15]．したがって，データの損失はどの企業にとっても許容できないものである．

課題

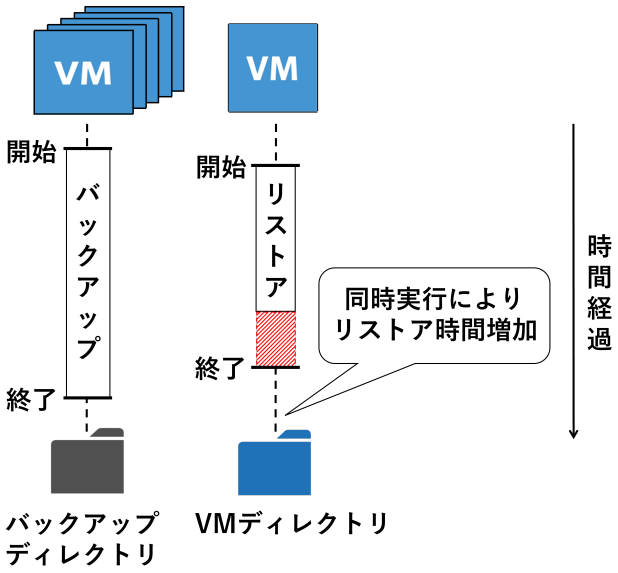

図 1: 課題の概要

本稿における課題の概要を図 1 に示す．課題は，バックアップを行っている最中にリストアを行うと，バックアップを行っていないリストア単体の場合と比較してリストア時間が増加することである．背景に述べたように，CDSL が運用している 9 台の PM がある．各 PM に作成されている VM のデータファイルは，ローカルネットワーク上に存在する Synology 社の Network Attached Storage(以下 NAS) 上にあるディレクトリに保存されている [16]．ここでは VM ディレクトリを指す．VM ディレクトリの中には複数台の VM が作成されている．バックアップは各 VM を 1 台ずつスナップショットを作成し，NAS 上にあるバックアップディレクトリにコピーしている．また，リストアを行う際にはバックアップディレクトリから VM のデータファイルをリストアしている．他の PM も同様に各ディレクトリに保存されている．

*2 https://x.gd/qQ9UI
*3 https://x.gd/gydBa

各章の概要

本稿は以下のように構成される．第 1 章では，本稿の背景と課題について述べる．第 2 章では，本稿の関連研究について述べる．第 3 章では，本稿での課題を解決するための提案手法について述べる．第 4 章では，提案した手法の実装について述べる．第 5 章では，評価実験の内容とその分析について述べる．第 6 章では，提案手法の議論を述べる．第 7 章では，本稿のまとめを述べる．

2. 関連研究

従来の VM のリストア技術では，チェックポイントイメージのサイズ削減に成功しているものの，復元時間の増加という課題がある．この問題に対処するため，新しい手法が提案されている [17]．この研究では，VM の全メモリ内容が完全に復元される前に VM をユーザが利用できるようにする方法を紹介している．従来の重複したメモリページを除外する技術に代わり，lazy fetch(遅延フェッチ) とアクセスインターセプト技術を用いることで，チェックポイントイメージのサイズを最適化しながらも効率的な復元を実現している．しかし，この手法は主に VDC 環境での仮想マシンの復元に焦点を当てており，他の環境や異なる種類のシステム障害に対する適用は限定的であるため本稿の課題を解決することができない．

従来のクラッシュ防止よりも短時間での回復を重視する設計が，低コストで高可用性のシステムを提供するために有効であることが示されている．この課題に対応するため，Sprite という UNIX に似た分散型オペレーティングシステムにおいて，リカバリーボックスを使用した回復手法が提案されている [18]．リカバリーボックスは，システムの重要部分を保存する安定したメモリ領域であり，エラー検出のチェックサムを使用し，復旧時にシステムを通常の再起動シーケンスに戻す．Sprite ファイルサーバの実装では，システムは 26 秒，データベースマネージャは 6 秒で復旧する．これにより，ユーザやアプリケーションがシステムダウンを意識しないほどの速さで復旧が可能である．ただし，この設計は Sprite システム専用であり，他のシステムには適用できないという欠点がある．

データ重複排除がバックアップシステムで広く使用されており，冗長データの量を削減する効果的な手法であることが示されている．元のバックアップデータを再構築するために，インライン重複排除の一般的な手順で復元が行われ，重複チャンクがさまざまなデータストリーム間で共有されている場合には読み取り増幅が生じる．この研究はバックアップシステムでの復元パフォーマンスを向上させ，高い重複排除率を保証する効果的なリージョンパーティ

ショニングベースの書き換えスキーム (ERP) を提案している [19]．ERP の重要なアイデアは，コンテナ間の冗長性情報を調査し，選択範囲を効果的に絞り込み，柔軟な数のコンテナを選択することである．しかし，この手法はデータ重複排除と復元において有用であるが，バックアップを行っている最中のリストア時間の増加をリアルタイムで防ぐという課題には対応していない．したがって，本稿の課題を解決することはできない．

ネットワーク機能仮想化 (NFV) は，ネットワーク機能を仮想マシンやコンテナ上に配置する技術であるが，一つの仮想ネットワーク機能 (VNF) に障害が発生すると，全体のネットワークが中断されることがある．高可用性を確保するために，プライマリ VNF のバックアップノードを設ける方法があるが，VNF の需要が考慮されていない．この研究は全体的な可用性需要を満たしながらバックアップの CPU やメモリの消費を最小限に抑えることを目的としている．この目的のために，問題の NP 困難性を証明し，解決するための微分進化にもとづく RABA-CDDE アルゴリズムを提案している．[20]．しかし，この研究は VNF チェーンの高可用性を確保し，バックアップの CPU やメモリの消費を最小限に抑えることであり，研究目的が違うため，本稿の課題を解決することはできない．

3. 提案

提案方式

本稿は，バックアップを行っている最中にリストアを行った際のリストア時間の増加を最小限に抑えることを提案する．ユーザはファイル転送に要する時間の 38 %を損失するとフラストレーションを感じる [21]．このデータから，各 VM のリストア時間によって，バックアップにおける CPU 使用量を制限し，リストア単体で行った時間に 38 %増加した時間までに完了させる．その後に制限していたバックアップの CPU 使用量を元の状態に戻すことでリストアを優先させる．これにより，ユーザのフラストレーションを軽減し，システムの運用効率を向上させることを目指す．特に，研究室のバックアップは 7:00 までに完了させる必要があるため，バックアップを停止することは避ける必要がある．そのため，バックアップを停止せずに CPU 使用量を制限してリストアを優先し，どちらも時間内に完了させる．

提案方式の流れを図 2 に示す．始めにバックアップを行っていない時にリストア単体で行った時間を測定し，基準値を設定する．基準値を設定した後，基準値の 38 %増加した時間を許容時間として設定する．次にバックアップを行っている最中にリストアが開始されたら，リストア時間をリアルタイムで監視し，許容時間以上かかるのかを判断する．判断方法はリストアが開始された後，一定間隔でリストアの進行状況を監視する．進行状況はパーセンテージで取得し，その時点での経過時間と合わせて予測完了時間を計算する．進行状況の取得間隔は 10 秒とし，例えば，リストア開始から 50 秒経過しその時点でリストアが 50 %完了している場合，予測時間は 100 秒となる．このようにして判断を行う．また，進行状況の取得間隔を 10 秒にした理由として，間隔が短すぎると，負荷がかかってしまい，リストアやバックアップを完了するまでの時間が増加してしまう．逆に間隔が長すぎると，CPU 使用量の制限を行うタイミングを逃し，リストア時間が許容時間を超えてしまう恐れがあるためである．許容時間以上かかると判断された場合，各 VM のリストア時間によって，バックアップおける CPU 使用量を変更し，許容時間までに完了させるという流れである．

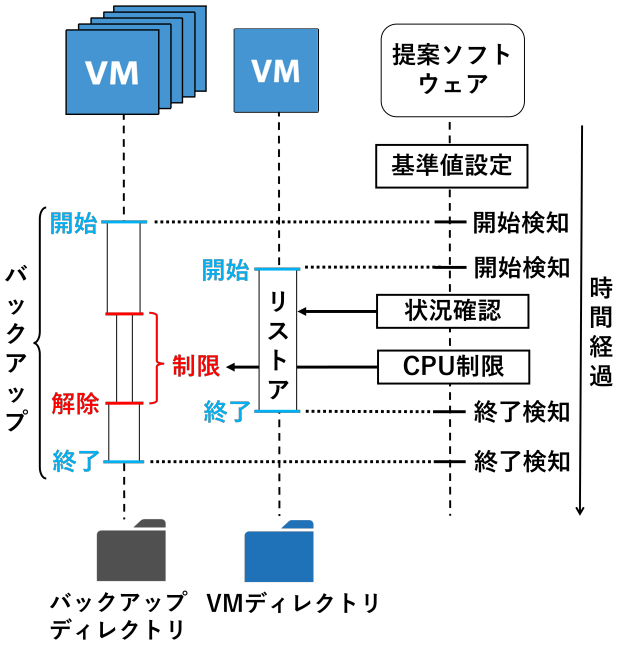

図 2: 提案方式の流れ

バックアップに使用している CPU 使用量の制限の計算は，VM のリストア時間によって異なるため，バックアップを行っていない時のリストア単体で行った時間をもとにして計算を行う．リストア単体で行った時間をもとにして CPU 使用量の制限する量を調整することで，リストア時間の 38 %以上の遅延を防ぐ．計算式を (1) に示す．

$$CPU_{\text{limit}} = CPU_{\text{core}} \times \left(\frac{T_{\text{current}}}{T_{\text{restore}} \times 1.38} \right) \qquad (1)$$

式の具体的な例をあげて説明する．許容時間内にリストアを完了するために必要な CPU 使用量の制限する量 CPU_{limit} として式から求める．T_{restore} は許容できるリストア時間を出すためにリストア単体で行った時間としている．その値に 1.38 倍したものが許容時間としている．

CPU_{core} はバックアップにおける CPU 使用量のことである．T_{current} はバックアップを行っている最中にかかっているリストア時間である．リストアしてからどれくらい経過したかを確認するために使用する．リストア単体で行った時間が 100 秒であるとする．この値に 1.38 倍したものが許容時間が T_{restore} である．現在のバックアップ進行中のリストア時間 T_{current} が 50 秒であり，CPU_{core} が現在の CPU 使用量が 4 コアであるとする．この場合，制限する CPU 使用量 CPU_{limit} は，約 1.05 コアである．

ユースケース・シナリオ

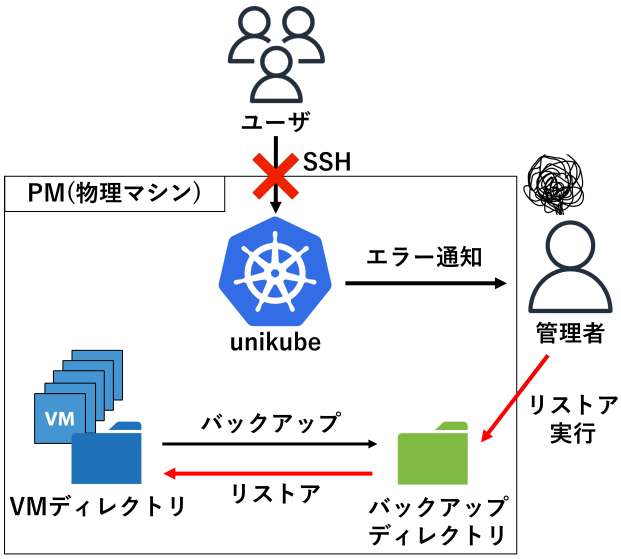

図 3: ユースケースシナリオ

本稿のユースケースとして CDSL での適用を想定している．ユースケースシナリオを図 3 に示す．CDSL には学生が誰でも使用できる Kubernetes 共有クラスター (以下 unikube) がある．unikube は現在，1 人の学生が管理者として運用している．以下に想定している場面として例を挙げる．運用している unikube が深夜に突然 SSH 接続できなくなってしまい使用できなくなってしまった．原因を調べるために直接 VM に入ろうとしたが入れず，コマンドが実行できなかった．SSH 接続と VM への直接アクセスも不可能であり，ログやシステム状態を確認することができなかった．そのため，再起動を行ったが，起動するために必要なファイルが破損していたため，正常に起動できなかった．これにより，リストアが必要になった．unikube は 4 人の使用者がいる．使用する理由として Kubernetes でクラスタを構築する際，VM の作成や K3s のインストールの工程を省略するためである．しかし，深夜に研究で unikube を使用していた時に突然 unikube のデータ損失や破損し，リストアしようとしたがバックアップと重なってしまいリストア時間が増加してしまい学生は研究ができなくなってしまうという状況を想定する．本稿の提案を使用することでバックアップを行っている最中に同時実行した時のリストア時間を短縮することができる．

4. 実装

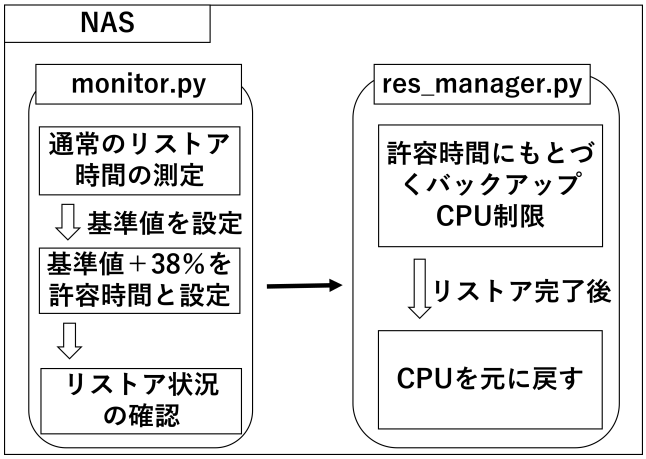

図 4: 提案ソフトウェアの概要

提案ソフトウェアの概要を図 4 に示す．提案ソフトウェアは monitor.py と res_manager.py の構成となっている．monitor.py では，バックアップを行っていない時にリストア単体で行った時間を測定し，基準値として設定する．その後，基準値の 38 %増加した時間を許容時間として設定する．その後，バックアップを行っている最中にリストアが行われた際の状況を一定間隔で進行状況を監視する．進行状況はパーセンテージで取得し，その時点での経過時間と合わせて予測完了時間を計算する．進行状況の取得間隔は 10 秒とし，バックアップにおける CPU 使用量を制限するかを判断する．res_manager.py はバックアップにおける CPU 使用量の制限する量を計算し，制限をかける．その後に制限していたバックアップの CPU 使用量を制限する前の状態に戻す．実装環境の構成は，CDSL において，ESXi 上に構築された仮想環境で複数の VM を運用する．リストア時間の測定のために，Python を用いてスクリプトを実装し，必要に応じてバックアップにおける CPU 使用量を制限する機能を持たせる．

5. 評価実験

評価は，提案手法が実際にバックアップを行っている最中にリストアを行った際のリストア時間の増加を短縮できるかどうかを確認する．まず，リストア単体の時間を測定するために，バックアップが行われていない単体でリストアを実行し，その所要時間を基準値として設定する．基準値を設定した後，基準値の 38 %増加した時間を許容時間として設定する．次にバックアップを行っている最中にリストアが開始されたら，リストア時間をリアルタイムで監視

し，許容時間以上かかるのかを判断する．許容時間以上かかると判断された場合，各 VM のリストア時間によって，バックアップにおける CPU 使用量を変更する．最終的にはリストア単体で行った時間に対して 38 %増加した場合にユーザのフラストレーションが高まるため，リストア時間が 38 %増加する前に完了させる．

基礎実験

基礎実験では，VM のバックアップを行っている最中にリストアを同時実行する場合とリストア単体で行った場合でリストア時間が変わるのかを CDSL の PM1 台を対象に基礎実験を各 8 回行った．

実験環境

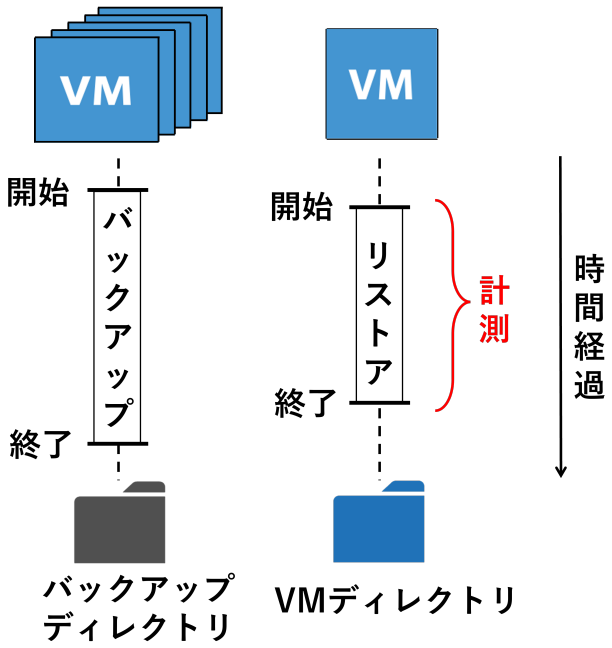

図 5: 基礎実験の環境

図 5 に基礎実験の環境を示す．対象の PM の中にある VM は合計 40 台あり，各 VM をバックアップしている最中にリストアを行う．バックアップを行うデータサイズは合計 1.15TB である．

基礎実験の結果

基礎実験を行った結果，図 6 はリストア単体で行った時間とバックアップとリストアを同時実行した時間を比較したものである．リストア単体は平均 163 秒で，同時実行は平均 790 秒であった．結果として，バックアップとリストアを同時実行することによってリストア時間は約 385 %増加した．

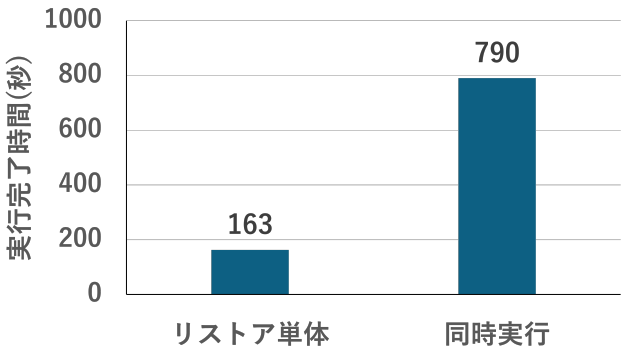

図 6: リストア単体と同時実行の比較

6. 議論

提案手法では，バックアップを行っている最中にリストアを開始した際に，リストア時間がリストア単体で行った時間の 38 %以上増加する場合にバックアップにおける CPU 使用量を制限し，リストアを優先させるようになっている．しかし，研究室で行っているバックアップは 3:00 から始まり，研究室は 7:00 から開くため，それまでにバックアップを完了させる必要がある．本稿の提案手法ではリストアの優先しか考慮していないため，バックアップについても考慮する必要がある．バックアップが時間内に完了しない程のリストアのデータサイズの場合にはリストア優先するためにバックアップにおける CPU 使用量を制限していたが，制限を解除し，バックアップを時間内に完了させる．バックアップは 7:00 までに完了させる必要があるため，リストアを優先するために提案手法で用いた予測時間を使用する．予測時間を超える場合と判断された場合，バックアップを優先させる．

また，バックアップにおける CPU 使用量を制限することでリストア時間が線形的になるかを実験する必要がある．実験として，複数の VM を用意し，バックアップを行っている最中にリストアを行い，バックアップの CPU 使用量の制限することで，リストア時間が線形的になるかを確認する．

7. おわりに

本研究の課題は，バックアップを行っている最中にリストアを行う際，バックアップを行っていないリストア単体の時間と比較してリストア時間が増加することであった．提案手法として，リストア単体で行った時間を基準値とし，基準値の 38 %増加した時間を許容時間として設定する．許容時間内にリストアを完了させるためにバックアップにおける CPU 使用量を VM のリストア時間によって制限する量を変更し，リストアを優先するソフトウェアを提案した．基礎実験として，リストア単体で行った時間とバックアップとリストアを同時実行した時間を比較した．CDSL

の仮想環境内のハイパーバイザ上に構築された 40 台の仮想マシンが作成されている PM を対象とした．その結果，バックアップとリストアを同時実行することによってリストア時間は約 385 %増加した．評価は，提案手法がバックアップを行っている最中にリストアを行った時間の増加を短縮するかを確認する．

参考文献

[1] Akbar, R., Husain, M. S. and Suaib, M.: Comparative study of various backup and monitoring techniques, *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 1530–1537 (online), DOI: 10.1109/ICGCIoT.2015.7380710 (2015).

[2] Mukherjee, P. and Salapura, V.: Challenges of DB2 Restore in a Distributed Systems Environment and Engineered Solutions, *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 38–42 (online), DOI: 10.1109/DSN-W.2018.00021 (2018).

[3] Xia, R., Yin, X., Alonso Lopez, J., Machida, F. and Trivedi, K. S.: Performance and Availability Modeling of ITSystems with Data Backup and Restore, *IEEE Transactions on Dependable and Secure Computing*, Vol. 11, No. 4, pp. 375–389 (online), DOI: 10.1109/TDSC.2013.50 (2014).

[4] Zhang, J. and Li, H.: Research and Implementation of a Data Backup and Recovery System for Important Business Areas, *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Vol. 2, pp. 432–437 (online), DOI: 10.1109/IHMSC.2017.209 (2017).

[5] Levina, A. and Semenov, A.: Jump-Based Backup: An Efficient Data Backup, *2024 13th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4 (online), DOI: 10.1109/MECO62516.2024.10577875 (2024).

[6] Xiao, W., Yang, Q., Ren, J., Xie, C. and Li, H.: Design and Analysis of Block-Level Snapshots for Data Protection and Recovery, *IEEE Transactions on Computers*, Vol. 58, No. 12, pp. 1615–1625 (online), DOI: 10.1109/TC.2009.107 (2009).

[7] Li, J., Liu, H., Cui, L., Li, B. and Wo, T.: iROW: An Efficient Live Snapshot System for Virtual Machine Disk, *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, pp. 376–383 (online), DOI: 10.1109/ICPADS.2012.59 (2012).

[8] Xiao, W., Yang, Q., Ren, J., Xie, C. and Li, H.: Design and analysis of block-level snapshots for data protection and recovery, *IEEE Transactions on Computers*, Vol. 58, No. 12, pp. 1615–1625 (2009).

[9] Yang, Y., Mao, B., Jiang, H., Yang, Y., Luo, H. and Wu, S.: SnapMig: Accelerating VM Live Storage Migration by Leveraging the Existing VM Snapshots in the Cloud, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, No. 6, pp. 1416–1427 (online), DOI: 10.1109/TPDS.2018.2790389 (2018).

[10] Jeong, W., Park, H. and Park, C.: KeyScrub: A Reliable Key Backup and Recovery Method for Blockchain, *IEEE Access*, Vol. 11, pp. 91747–91755 (online), DOI: 10.1109/ACCESS.2023.3308208 (2023).

[11] Paul, S., K.A., S. and Abraham, J. P.: Block Level Incremental Backup With Brotli Compression, *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, pp. 1–5 (online), DOI: 10.1109/ICCSDET.2018.8821114 (2018).

[12] Levitin, G., Xing, L., Zhai, Q. and Dai, Y.: Optimization of Full versus Incremental Periodic Backup Policy, *IEEE Transactions on Dependable and Secure Computing*, Vol. 13, No. 6, pp. 644–656 (online), DOI: 10.1109/TDSC.2015.2413404 (2016).

[13] Song, Y., Liu, K., Gu, Y., Jiang, Y., Sun, W. and Fang, L.: Data Resilience Protection for Power Data Center Based on Automatic Identification of Key Data and Incremental Backup, *2023 International Conference on Intelligent Management and Software Engineering (IMSE)*, pp. 56–59 (online), DOI: 10.1109/IMSE61332.2023.00018 (2023).

[14] Javaraiah, V.: Backup for cloud and disaster recovery for consumers and SMBs, *2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS)*, pp. 1–3 (online), DOI: 10.1109/ANTS.2011.6163671 (2011).

[15] Du, J., Yu, H. and Zheng, W.: MassStore: A low bandwidth, high De-duplication efficiency network backup system, *2012 International Conference on Systems and Informatics (ICSAI2012)*, pp. 886–890 (online), DOI: 10.1109/ICSAI.2012.6223150 (2012).

[16] Nemoto, J., Sutoh, A. and Iwasaki, M.: File System Backup to Object Storage for On-Demand Restore, *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 946–952 (online), DOI: 10.1109/IIAI-AAI.2016.116 (2016).

[17] Egger, B., Gustafsson, E., Jo, C. and Son, J.: Efficiently restoring virtual machines, *International Journal of Parallel Programming*, Vol. 43, pp. 421–439 (2015).

[18] Baker, M.: The recovery box: Using fast recovery to provide high availability in the UNIX environment, *In Proceedings USENIX Summer Conference*, Citeseer (1992).

[19] Liu, W., Lu, Y., Wu, C., Li, J. and Guo, M.: ERP: An Efficient Rewrite Scheme to Improve the Inline Deduplication Restore Performance in Backup Systems, *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 371–378 (online), DOI: 10.1109/ICPADS56603.2022.00055 (2023).

[20] Zhang, J., Wang, Z., Peng, C., Zhang, L., Huang, T. and Liu, Y.: RABA: Resource-Aware Backup Allocation For A Chain of Virtual Network Functions, *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1918–1926 (online), DOI: 10.1109/INFOCOM.2019.8737565 (2019).

[21] Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J. and Shneiderman, B.: Determining causes and severity of end-user frustration, *International journal of human-computer interaction*, Vol. 17, No. 3, pp. 333–356 (2004).

