ストレージ管理標準仕様を用いた大規模環境向け構成情報収集方式の提案

坂下幸徳† 河野泰隆† 柴山司† 中島淳† 敷田幹文††

近年，データセンタ内のストレージ環境が大規模化と共に．複数ベンダや新旧のストレージ装置といった様々な機種が混在するヘテロジニアス環境化が進んでいる．さらに，災害対策のディザスタリカバリやストレージ仮想化などの複数ストレージ装置を連携させた運用が一般化してきた．このように大規模複雑化するストレージ環境を管理するためには，複数ストレージ装置の構成情報を一元管理することが不可欠である．従来のストレージ管理ソフトウェアは，ストレージ管理標準仕様である SMI-S(Storage Management Initiative-Specification)を活用し，ヘテロジニアス環境のストレージ装置の構成情報の一元管理が可能であるが，ストレージ装置が 100 台近くある大規模な環境に対しては，ストレージ装置から構成情報を一括収集する時間が 10 時間以上必要であり，更に使用するメモリ量が 5GBytes 以上必要になってしまい運用出来ない．本論文では，SMI-S のデータモデルに着目し並列処理化と使用メモリ量の削減可能なオペレーションを組み合わせることで，大規模環境でも運用に耐えうる構成情報収集方式を提案する．また，本提案方式の試作プログラムを使い測定した結果を用い，本提案方式の有効性について議論する．

A Gathering Method of Storage Resource Information by Standard Specification Use for Large-scale Environment

YUKINORI SAKASHITA† YASUTAKA KONO†
TSUKASA SHIBAYAMA† JUN NAKAJIMA†
MIKIFUMI SHIKIDA††

In these years, the storage environment in the data center becomes large-scale, and the heterogeneous that is the mixture environment of some vender, old and new storage systems. In addition, a solution of disaster recovery of measures to deal with natural calamities and a function of storage virtualization has generalized in the storage environment. It is necessary to unitary manage resource information on the two or more storage system to manage such a storage environment. A current storage management software can unitary manage resource information on the storage system of the heterogeneous environment by using standard specification(SMI-S).However a large-scale environment with 100 storage systems or more can't be managed. Because time to gathering resource information from the storage system is 10 hours or more necessary, and the used memory is necessary for 5GBytes or more. This paper proposes the gathering method of storage resource information which can be used even in a large-scale environment. The method is to combine the parallel processing by the data model SMI-S with the new operation which can be reduced the used memory. We illustrate the method with a result of the measurement using the prototype system and discuss is usefulness.

1. はじめに *

近年，企業や教育機関が所有するデータセンタへストレージ装置を導入し，保有するデータを一元管理する傾向がある．

このようなデータセンタでは，近年の急速なデータ量の増加やストレージ装置自身が備える機能の拡充にともないストレージ装置の買い増しを年々行っている．その結果，所有するストレージ装置の台数が増加しており，世界中での全出荷台数が年率約 4 倍のペースで増加していることを考えると，約 5 年後には 100 台以上ものストレージ装置を所有するデータセンタの登場が予測される[1]．さらに，買い増しを行う際，他のベンダからの購入や新製品の導入により，データセンタ内のストレージ装置は，複数ベンダや新旧の様々なストレージ装置が混在するヘテロジニアス環境(以下，ヘテロ環境と略す)になっている．

また，他の傾向としては，ストレージ装置単体ではなく複数ストレージ装置を連携させた運用が普及してきている．これは，ストレージ装置に格納されるデータを地震・火災・テロといった災害から保護すべく，地理的に離れたデータセンタ間でデータ保護を行うディザスタリカバリや，複数ストレージ装置を接続し，あたかも 1 台の巨大なストレージ装置と見せるストレージ仮想化が登場してきたためである．

このような状況の中，大規模複雑化するストレージ環境を一元管理し管理コストを削減できるストレージ管理ソフトウェアが必要不可欠となっている．

これに対し，従来のストレージ管理ソフトウェアでは，まず，複雑化の一因であるヘテロ環境向けに業界団体である SNIA(Storage Networking Industry Association)が策定しているストレージ管理標準仕様 SMI-S（Storage Management Initiative-Specification）[2]

*† (株)日立製作所 システム開発研究所
Hitachi Ltd., Systems Development Laboratory
†† 北陸先端科学技術大学院大学情報科学センター
Center for Information Science, Japan Advanced Institute of Science and Technology

を利用することで，ヘテロ環境への対応を行っている．しかし，これらは，ストレージ環境の構成情報を収集する際のスケーラビリティが弱く．大規模化する環境に対応出来なかった．

そこで，本論文では，SMI-S を用いたストレージ管理ソフトウェア向けに大規模なストレージ環境を対象としたスケーラビリティを向上した構成情報の収集方式を提案する．

以下，2 章では，従来のストレージ管理ソフトウェアについて述べ，3 章では本論文で対象とする問題について述べる．4 章では問題解決のアプローチを提案し，5 章ではこれに従った方式と試作システムを説明する．6 章ではこれを用いた測定を行い，7 章で測定結果に基づいた提案方式の有効性を議論する．

2. 関連研究

本章では，SMI-S および SMI-S に関するこれまでの研究および商用のストレージ管理ソフトウェアに関して述べる．

2.1 SMI-S の概要

SMI-S はベンダ毎に異なるストレージ装置や FC スイッチなど SAN 環境の機器の管理 I/F(Inter Face)を統一し，ヘテロ環境における管理を統一のものにすべく SNIA が策定した国際標準(ANSI,ISO/IEC)の管理 I/F である．2003 年に SMI-S1.0 が公開され，2009 年には SMI-S1.4 が公開されている．SNIA の報告[3]によると，2010 年の時点で，SMI-S を採用する装置の総数は 800 以上あり，エンタープライズストレージ市場における大手ベンダの 90%以上の製品が採用している．

また，SMI-S はその仕様に，データモデルとして DMTF(Distributed Management Task Force, Inc.)が策定している CIM(Common Information Model)[4]を採用している．更に，その通信プロトコルには，CIM を XML フォーマットに変換し，HTTP 上でデータ通信を行う規格である WBEM(Web-Based Enterprise Management)[5]を採用している．

SMI-S を利用したシステムでは，Provider と呼ばれる SMI-S に準拠したストレージ装置などの機器が仕様に準拠したデータモデルで構成情報を提供し，Client と呼ばれる管理ソフトウェアが，Provider から情報を収集することで管理を行っている．

2.2 SMI-S 活用のストレージ管理ソフトウェア

2.1 節で述べた SMI-S を，ヘテロ環境のストレージ装置を管理すべくストレージ管理ソフトウェアへ適用する研究[6][7][8]が行われている．また，SMI-S を活用した商用のストレージ管理ソフトウェア[9][10][11]も登場してきている．

いずれの製品も，複数装置の構成情報を一元管理することを特徴としており，図 1 に示すモジュールで構成されている．これらストレージ管理ソフトウェアは，まず，ストレージ装置等と構成情報収集モジュールを通じ通信を行うことで各装置の構成情報を収集する．収集した構成情報は構成情報 DB(Database)に保存された後，GUI/CLI を通じストレージ装置の情報をストレージ管理者へ提示し運用管理を行っている．この構成情報収集モジュールに SMI-S を適用することで，ヘテロ環境の管理を実現している．

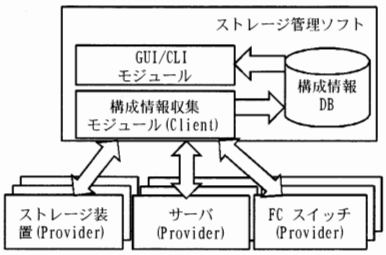

図 1 ストレージ管理ソフトウェアの構成図
Fig 1 Component of Storage Management Software

3. 大規模環境の一元管理における問題点

本章では，ストレージ環境の大規模化に伴い発生するストレージ管理ソフトウェアの問題点について述べる．

3.1 情報収集時間の増大

まず，一つ目の問題点としてストレージ装置から構成情報を収集する際の時間の増大があげられる．

これの原因として考えられるのは，ストレージ管理ソフトウェアで複数台のストレージ装置を一元管理しなければならなくなったことと，データセンタ内のストレージ装置の台数増加である．ディザスタリカバリやストレージ仮想化により，複数ストレージ装置を跨って運用するケースが増えたことで，ストレージ管理ソフトウェアは，障害解析や適切なリソース配分といった管理を行うために，複数ストレージ装置の構成情報を一元管理しなければならない．また，単に一元管理するだけでなく，適切な管理を行うためには複数のストレージ装置間で構成情報の整合性を保つことが不可欠であり，ストレージ装置間の構成収集の時間差が重要である．つまり，ストレージ装置により構成情報を収集した時刻に時間差が大きいと，正しい構成情報とならず，適切な管理が出来ない．

しかし，データセンタ内のストレージ装置の台数が増加していることで，管理対象のストレージ装置から構成情報を収集する時間も増大している．

たとえば，100 台以上のストレージ装置から構成されるような大規模環境において，ストレージ装置の利用が少ないと想定される夜間に構成情報の一括収集を行う運用を想定した場合を考える．この場合，従来方式では，一晩(7 時間)かかっても構成情報の収集が完了しないだけでなく，最初に収集したストレージ装置と最後に収集したストレージ装置では 10 時間以上もの時間差が生じてしまう．そのため，情報の整合性を保つことが出来ず実運用に耐えられない．

3.2 情報取得時におけるメモリ使用量の増大

二つ目の問題点として，ストレージ装置から構成情報を取得する際のメモリ使用量の増大があげられる．

これの原因として考えられるのは，一台のストレージ装置が備える HDD，Port，Volume といったリソース

数の増加である．特にエンタープライズクラスのストレージ装置では，1 台のストレージ装置が備える Volume 数が増加しており，最大構成で 128,000 Volume を備えるストレージ装置も登場し，最大構成にて利用するユーザも存在している．そのため，構成情報を収集する場合において，Volume など数が多いリソースの構成情報を収集する場合にメモリ量が問題となる．

構成情報を収集する際の通信プロトコルである WBEM では，指定されたリソースの情報すべてを 1 つの XML データにエンコードする．そのため，1 回の通信で送信されるデータ量が巨大化するだけでなく，その XML データの生成処理，解析処理に膨大なメモリが必要となる．一例をあげると，Volume を表現する CIM_StorageVolume の 1 インスタンスあたりのデータ量は約 11Kbytes である．1 台のストレージに 10,000 Volume 存在する場合では，約 100Mbytes ものデータ量となる．これを，32bit 環境の Java1.5 の標準の XML パーサを利用した場合のメモリ使用量を測定すると，XML データの読み込み時に Java のオブジェクトに変換するため，XML データの読み込み処理だけで，約 500Mbytes 必要となる．また，128,000Volume を持つストレージ装置の場合では，約 1.0GBytes ものデータ量となり，これのデータ処理には 5.0GBytes 以上ものメモリが必要となる．これは，32bit 環境の Java VM のメモリのヒープサイズが理論値で最大 2GBytes であることを考慮すると，同環境では，このような大規模なストレージ装置 1 台すら管理出来ない．

メモリ量に関しては，ストレージ管理ソフトウェアを導入する環境を 64Bit 環境で且つ大量のメモリを搭載した管理用 PC に導入するなど，ハードウェア側からのアプローチにより改善する方法もある．しかし，ストレージ装置の大規模化が進んでいる状況を考慮すると，ハードウェア側からのアプローチだけでは，限界があり，ストレージ管理ソフトウェアでのメモリ量の削減が必要である．

4. 新オペレーション方式と並列処理化の提案

本章では，前章で述べた問題点に対する解決に向け，情報処理取得時におけるメモリ使用量の削減を行い，更に，削減したメモリを使い並列処理を行うことで収集時間の短縮を行うアプローチを提案する．

4.1 情報取得時におけるメモリ使用量の削減

メモリ使用量の削減に向け，WBEM の“CIM Operations over HTTP 1.3.1”で採用された新オペレーションである Pulled Enumeration Operations を適用することでメモリ使用量の削減を狙う．

この新オペレーションは，Client からのリクエストを受けると，Provider から指定されたインスタンス数毎の XML データに分割し，レスポンスを返す．これにより，1 回の通信で送信されるデータ量を削減でき，メモリ量の削減が見込める．

また，別のアプローチ案として，SMI-S1.4 までに規定されている既存オペレーションを使う方法が考えられる．既存オペレーションで利用される XML フォーマットの仕様上，全ての XML データを受信しなければ，XML データとしての最低限のフォーマット規約を満たせないため，Java の XML パーサで処理出来ず，全ての XML データを受信した後にパースしなければならなかった．そのため，全インスタンスの情報を 1 度に取得する EnumerateInstances のようなオペレーションでは，3.2 節で例にあげた 1 台のストレージに 10,000Volume 存在する場合，CIM_StorageVolume クラスに対する 1 回のリクエストで約 100M のデータ量となってしまい，メモリ消費の主要因となってしまう．そこで，メモリを大量消費するオペレーションである EnumerateInstances オペレーションらを使わず図2に示すように，これをメモリ消費の少ない別オペレーションに分割することでメモリ量を減らす案を考える．

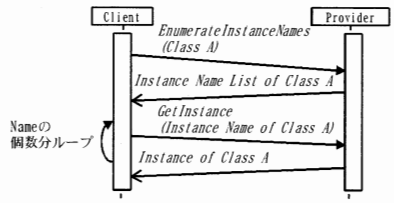

図 2 EnumerateInstances の代替手段
Fig 2 Alternative of EnumerateInstances operation

まず，最初のオペレーションとして Class A のインスタンス名の一覧を取得する EnumerateInstanceNames オペレーションを実行する．次に取得したインスタンス名を使い，1 インスタンスの情報を取得するオペレーションである GetInstance オペレーションを実行し，インスタンスを取得する．これを，EnumerateInstanceNames オペレーションで取得したインスタンスの数分実行すれば，EnumerateInstacens オペレーションと同様の結果を得ることが出来る．

しかし，このアプローチ案には 2 つの問題がある．

一つ目の問題点は，実行時間が増大してしまう点である．EnumerateInstances オペレーションであれば 1 回の通信であったが，EnumerateInstanceNames オペレーションと GetInstance オペレーションに分割したことで，1+Ni 回(Ni:インスタンス数)の通信が発生し，実行時間が増大してしまう．

二つ目の問題点は，EnumerateInstanceNames オペレーションであっても将来ストレージ装置が大量にリソースを備えた場合に対応できなくなる点である．

EnumerateInstanceNames のオペレーションは，EnumerateInstances オペレーションとは異なり，インスタンス名のみ取得するオペレーションであるため，約 1/10 にデータ量を削減出来る．しかし，大量リソースを抱えたストレージ装置が今後登場した場合に，データ量が増加しメモリ不足となる．

そのため，既存オペレーションの組み合わせではなく，新オペレーションである Pulled Enumerateion Operations を採用する．

4.2 情報収集時間の短縮

情報収集時間の短縮に向け，ストレージ管理ソフトウェア(Client)からストレージ装置(Provider)へのリクエストを並列化することを考える．

しかし，SMI-S では，クラス間の依存関係が強く，単純に並列化することが出来ない．例をあげると，SMI-S では図 3(a)に示す Class A 及び Class B の情報(インスタンス)を取得する場合，図 3(b)のシーケンスのように，まず EnumerateInstances オペレーションにて Class B のインスタンスを取得し，次に，Associators オペレーションに，先に取得した Class B のインスタンスを入力値として渡すことで，渡された Class B のインスタンスに関連した Class A のインスタンスを取得する．

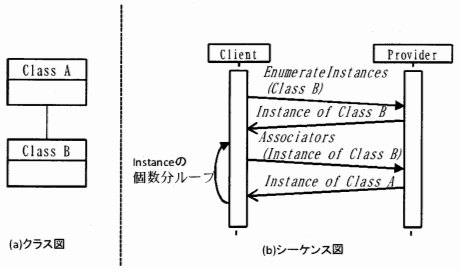

図 3 SMI-S における情報取得シーケンス
Fig 3 Gathering Sequence of SMI-S

つまり，Class B のインスタンスを取得しなければ Class A の情報が取得できないというクラス間の依存関係が存在する．このように SMI-S ではクラス間の依存関係があるため，本アプローチを取るためには，この依存関係を考慮した並列処理を考案する必要がある．

また，本アプローチでは，処理を並列化することで，並列数に応じ使用するメモリ量が増加する．

そのため，並列数も無制限にあげるのではなく，4.1 節にて削減したメモリ量に応じ，並列数を調整しなければならない．

5. 大規模環境向けストレージ構成情報収集法

本章では，前章で述べたアプローチに従い，試作したシステムについて述べる．

5.1 概要

4 章で示したアプローチを実践するために，次の試作を行う．

(1) SMI-S の各クラスの依存関係を考慮した並列処理化の開発
(2) 新オペレーション Pulled Enumeration Operations 採用したライブラリ利用

(1)に関しては後述の 5.2 節にて述べる．(2)に関しては，Pulled Enumeration Operations を採用している WBEM のライブラリ(J WBEM Server3.1.2)を採用することで対応する．図 4 にシステム構成図を示す．開発言語には Java を使用した．

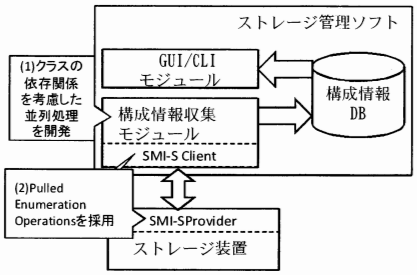

図 4 システム構成図
Fig 4Component of System

5.2 リソースの依存関係に基づく並列化方法

次に，SMI-S の各クラスの依存関係を考慮した並列処理について述べる．

SMI-S では，3.2 節で述べたように，ストレージ装置のリソースを表す各クラスの依存関係があり，ストレージ装置の構成情報の収集処理を単純には並列化できない．そのため，これを並列化するために，まずリソースの依存関係に着目し，依存関係の強いクラス群をグループとしてまとめ，分類した．

グループにまとめる際に，SMI-S では，ストレージ装置の各リソースをそのモデルの意味から，Profile というカテゴリにて分類し定義していることに着目した．図 5 及び表 1 にグループ化の一部例を示す．

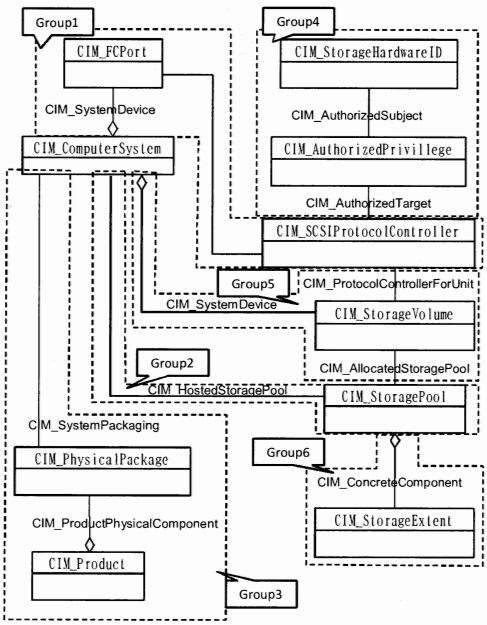

図 5 取得対象クラスのグループ化例
Fig 5 Group of Target Classes

表 1 SMI-S Profile と Group の対応
Table 1Relationship between SMI-S Profile and Group

SMI-S Profile 名	対応する Group
FC Target Ports	Group1
Block Services (Pool Manipulation Capabilities, and Settings)	Group2
Physical Package	Group3
Masking and Mapping	Group4
Block Services (LUN Creation)	Group5
Disk Drive Lite	Group6

()は，SMI-S1.0 での名称

グループにまとめた結果，各グループ間で構成情報を取得する際に，取得順番に依存関係があることが判明した．そこで，さらに，上記で分けたグループ間での依存関係に着目したところ表 2 のように分類されることが判明した．

表 2 グループ間での依存関係
Table 2 Depend Relation between Groups

グループ間に依存なし	Group1, Group2, Group3
1 つのグループに依存	Group4, Group6
複数グループに依存	Group5

そこで，依存関係のないグループに関しては，単純に並列化し，依存関係のあるグループに関しては，依存元のグループの情報が取得した後，情報取得することとした．図 6 に構成情報取得における並列化処理をアクティビティ図にて示す．

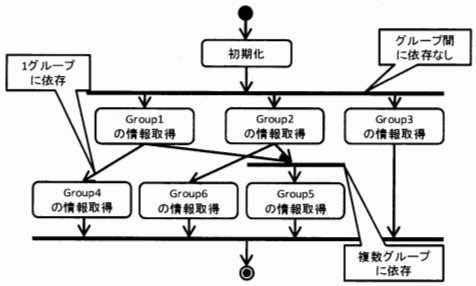

図 6 情報取得の並列化
Fig 6 Parallel Sequence of Getting Method

上述のように各クラスを依存関係の強い単位でグループにまとめし，さらにグループ間の依存関係を考慮し並列化した．これにより，ストレージ装置の構成情報の収集処理を並列化した．なお，筆者らが試作したシステムでは，SMI-S に規定されたクラスを対象に，全 51 クラスに対し，上記方法にて並列化処理を行った．

6. 測定

本章では，5 章で述べた試作システムを使い測定した，構成情報収集時間，使用メモリ量の結果を述べる．

6.1 測定環境

5 章で述べた試作システムを使い，大規模環境で問題となる構成情報収集時間及び使用メモリ量をそれぞれ測定した．表 3 に測定環境を示す．

表 3 測定環境
Table 3 Measurement environment

ストレージ管理ソフトウェア (試作システム)	PC	CPU:Core2Due 2.00GHz Memory:4GBytes
	OS	Windows XP Professional SP3
管理対象ストレージ装置	A 社製	Volume 数 3200
	B 社製	Volume 数 3100
	C 社製	Volume 数 3050

6.2 構成情報収集時間に関する測定結果

5 章の試作システムを使い，最も効果的な並列数(スレッド数)を調査すべく，主要ストレージベンダ 3 社に対し測定を実施した．なお，測定に利用した試作システムでは，5.2 節で示したクラスを対象に，5 種類のオペレーションを組み合わせ情報取得している．結果を図 7 に示す．

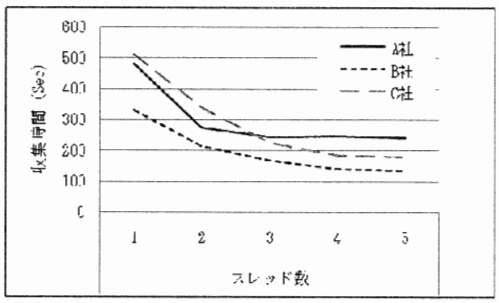

図 7 構成情報取得時の収集時間とスレッド数
Fig 7 Getting Time and Number of Thread

主要ストレージベンダ 3 社のストレージ装置に対し測定した結果によると，スレッド数 1 から 2 にかけて，並列処理による効果が高く各社ともスレッド数 4 で並列処理の効果が収束傾向となっている．

次に，メモリ量削減を目的に採用した Pulled Enumeration Operations のインスタンス取得にかかる取得時間について測定を行った．測定は，ストレージ管理ソフトウェアで利用されているオペレーションのうち 80%以上を占める EnumerateInstances, Associators, References の 3 つの従来のオペレーションに関し，これに対応した Pulled Enumeration Operations を対象とした．測定には，C 社ストレージ装置を用い，ストレージ装置の構成を，1000 インスタンス，2000 インスタンス，4000 インスタンスとなるよう Volume 数を変更し，測定を実施した．さらに，測定では，Pulled Enumeration Operations のオペレーション OpenEnumerateInstances，OpenAssociatorInstances，OpenRefrenceInstances の 3 つに対し，それぞれ 1 回のリクエストで取得するインスタンス数(MaxObjectCount)を指定可能な最大数である 1000 から 100，50 と変化させ構成情報の収集時間の測定を行った．結果を図 8，図 9，図 10 に示す．

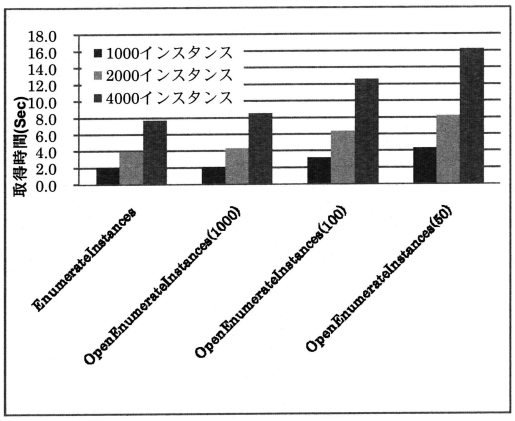

図 8 OpenEnumerateInstances の取得時間
Fig 8 Getting Time of OpenEnumerateInstances operation

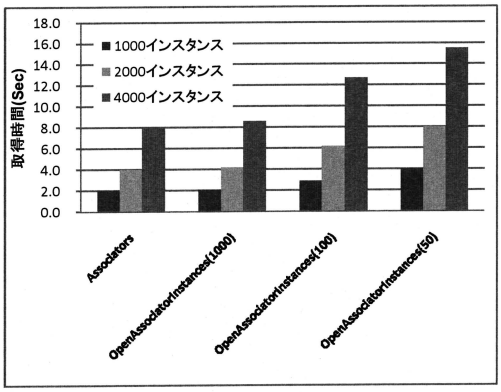

図 9 OpenAssociatorInstances の取得時間
Fig 9 Getting Time of OpenAssociatiorInstances operation

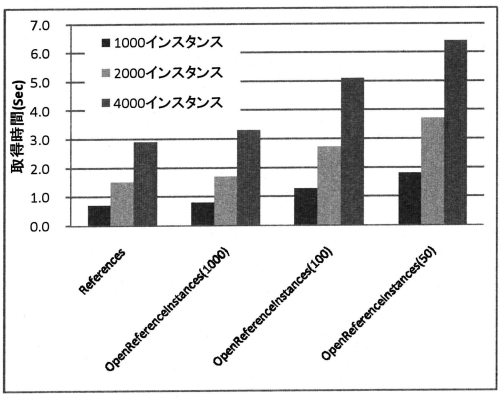

図 10 OpenRefrenceInstances の取得時間
Fig 10 Getting Time of OpenRefrenceInstances operation

測定の結果より，MaxObjectCount が 1000 の場合をみると，従来のオペレーション（EnumerateInstances，Associators，References）と比較し Pulled Enumeration Operations 採用による情報取得時間は 1 秒以下と小さい．これにより，Pulled Enumeration Operations にて MaxObjectCount に応じ，XML データを分割するという新規処理に対する性能の影響は少ないと言える．

6.3 メモリ量に関する測定

次に，C 社ストレージ装置を用い，Pulled Enumeration Operations を利用した場合の図 4 の構成情報収集モジュールの使用メモリ量を測定した．測定では，Pulled Enumeration Operations の各オペレーションの MaxObjectCount を 1000 とした．

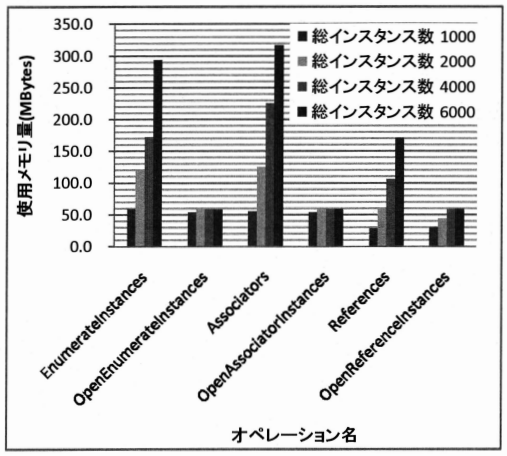

図 11 Pull Enumeration Operations 使用メモリ量
Fig 11 Using memory size of Pull Enumeration Operations

測定の結果，EnumerateInstances，Associators，References の旧オペレーションでは，インスタンス数に比例し，使用メモリ量が増加しているのに対し，Pulled Enumeration オペレーションでは，最大 56Mbytes で，使用メモリ量が一定となっている．また，OpenReferenceInstances オペレーションにおいては，インスタンス数が 1000-4000 の間，使用メモリ量が単調増加している．これは，OpenReferenceInstances オペレーションで取得する情報が，他のオペレーションに比べ 1 インスタンスあたりのデータ量が小さいため，56Mbytes まで利用していないためと考える．

7. 議論

本章では，6 章の測定結果から，大規模なストレージ環境における提案方法の有用性について議論を行う．

7.1 情報収集の並列化の有用性

従来は，3.1 節，4.2 節で述べたように SMI-S でモデル化されているストレージ装置のリソースを示すクラスにて，各クラス間の依存関係があることから，構成情報の収集を並列化することが困難であった.しかし,，提案方式では，ストレージ装置のリソースを示す各クラスの依存関係を考慮しグループ化を行うことで，並

列処理を可能とした．

この並列処理について，構成情報の取得時間を測定した結果，6.2 節に示すように，並列処理のスレッド数 4 が最も並列処理の効果が高く，それ以上スレッド数を増やしても効果が少ないことが判明した．これは，グループ間の依存関係があるため，スレッド数 4 で収束傾向となったと考えられる．しかし，提案方法により Volume 数が約 3000Volume の規模のストレージ装置から構成情報を収集するのに，並列化しない従来方式だと約 7 分(441 秒)かかっていたものが，提案方式では，約 3 分(191 秒)になり，約 57%もの時間短縮に成功した．これは，複数台を管理するケースにおいて，順次構成情報を取得する場合，N 台のストレージ装置から構成情報を収集するのに，441N 秒必要だったものが 191N 秒に短縮出来る．

次に，対応できるストレージ環境の規模の観点から述べる．ストレージ装置からの構成情報収集を一晩(7 時間と想定)で完了しようと計画した場合，従来方式だと 44N 秒必要であったため，57 台まで扱えなかった．これに対し，提案方式では 191N 秒と短縮したことで，131 台まで管理出来るようになる．

つまり，ストレージ装置の構成情報の収集処理を並列化したことで，ストレージ装置に対する収集時間を短縮でき，従来方式の 2 倍以上のストレージ装置の台数を管理出来るようになることから，本提案方式は有用である．

7.2 Pulled Enumeration Operations の有用性

次に，使用メモリ量の削減のために導入した Pulled Enumerateion Operation について述べる．3.2 節，4.2 節で述べたように，従来方式では，取得するインスタンス数に比例し使用メモリ量が増加していた．これに対し，提案方式で採用した Pulled Enumeration Operations では，その設計思想通り，取得するインスタンス数が増加しても，使用メモリ量は一定で抑えられており，最大約 56Mbytes のメモリ量となることが 6.3 節の結果，実証された．これは，測定した Pulled Enumeration Operations の 3 つのメソッド共すべて同一の傾向である．

これにより，従来方式で Volume の情報を収集する場合，32bit 環境の Java の環境下では，1 台のストレージ装置あたり約 20,000 Volume のリソースがあると，メモリ不足となり処理の限界であったが，提案方式では，これらの限界がなくなったと言える．つまり，1 台のストレージ装置で 20,000 Volume 以上ものリソースを標準で備えるエンタープライズクラスのストレージ装置を管理する場合であっても，提案方式を用いれば，ストレージ管理ソフトウェアの使用メモリ量を抑えることが出来る．

また，Pulled Enumerateion Operations を採用したことで，デメリットとなる情報取得時間について，6.2 節で測定を行った結果，従来の EnumerateInstances オペレーションらと比較し，性能劣化はわずかであり影響は少ない結果であった．

また，Pulled Enumeration Operations を採用する場合に，一度に取得するインスタンス数の指定(MaxObjectCount)の値が構成情報の取得時間に影響する．そこで 6.2 節の測定結果より，近似式をもとめたところ表 4 のようになった．

表 4 新オペレーションの測定による近似式
Table 4 Approximate formula of New operations

オペレーション名	MaxObject Count	近似式
OpenEnumerateInstances	50	$y = 2.2e^{0.7x}$
	100	$y = 1.6e^{0.7x}$
	1000	$y = 1.1e^{0.7x}$
OpenAssociatorInstances	50	$y = 2.0e^{0.7x}$
	100	$y = 1.4e^{0.7x}$
	1000	$y = 1.0e^{0.7x}$
OpenReferenceInstances	50	$y = 1.0e^{0.6x}$
	100	$y = 0.6e^{0.7x}$
	1000	$y = 0.4e^{0.7x}$

y:取得時間(Sec)，x:取得インスタンス(リソース)数

表 4 に示すように MaxObjectCount を 1000 がいずれのオペレーションでも係数が最も小さくなっている．特に，ストレージ仮想化など高度な機能の登場により，ストレージ装置の構成が複雑になり，リソース数が増加傾向にある．そのため，管理するリソース数の増加による処理時間への影響を少なくするため，係数が最も小さい MaxObjectCount を 1000 と設定するのが好ましいと考える．

上記より，Pulled Enumeration Operations の採用により，メモリ不足により制約を受けていた管理リソースの制約を排除でき，さらに処理時間への影響も少ないことから，本提案方式は有用である．

7.3 提案方法の大規模環境における有効性

7.1 節，7.2 節の結果より情報収集の並列化，Pulled Enumeration Operations がそれぞれ情報取得時間の短縮，使用メモリ量の削減に各々有用であることが判明した．

さらに，これらを組み合わせた場合を考える．まず，1 台のストレージ装置からの情報収集に関しては，7.1 節で述べたように，スレッド数 4 で処理した場合，各社の平均をとると 191(sec)である．この場合の使用メモリ量は，Pulled Enumeration Operations を利用することで，1 スレッドあたり最大 56Mbytes となることから，スレッド数 4 で最大 224Mbytes となる．

次に 7.1 節では，1 台のストレージ装置に対する並列化処理を考えたが，本節では，さらに情報収集の処理時間の短縮を狙い図 12 に示すように複数台のストレージ装置から同時に情報収集することを考える．

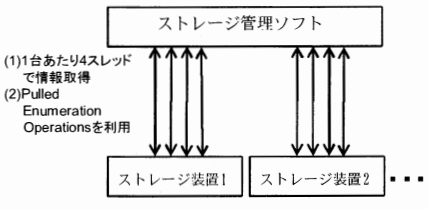

図 12 複数台のストレージ装置からの情報収集
Fig 12 Getting Method from the plural Storage

SMI-S のモデルは，ストレージ装置単位でモデルが独立しているため，複数台のストレージ装置に対し，同時に情報収集が可能である．つまり，ストレージ装置 1，ストレージ装置 2 と 2 台のストレージ装置から情報を収集する場合，合計 8 スレッドでの情報収集できる．

この複数台のストレージ装置から情報収集する場合の使用メモリ量と情報収集時間を式(1)式(2)にて示す．

$$Um = 224Nth \quad (1)$$

$$Rt = 191\frac{N}{Nth} \quad (2)$$

Um:使用メモリ量(Mbytes), Nth:同時に情報収集するストレージ台数，Rt:情報収集時間(Sec)，N:管理対象とするストレージ装置の全台数

たとえば，32bit 環境の Java VM に，ストレージ装置からの構成情報収集処理に 1Gbytes のメモリを割り当てたストレージ管理ソフトウェアにて，100 台のストレージ装置を管理すると仮定する．その場合，式(1)式(2)にあてはめ算出すると，同時に構成情報収集可能なストレージ装置の台数は 4 台となり，構成情報取得時間は，約 1 時間 20 分(4775 Sec)ですべてのストレージ装置に対する構成情報収集が可能となる．これは，従来方式では，約 12 時間以上(44133 Sec)必要であった情報収集の処理時間を約 90%短縮出来る．

次に，対応できるストレージ環境の規模の観点から述べる．ストレージ装置からの構成情報収集を一晩で完了しようと計画した場合，構成情報が収集能なストレージ装置の台数も従来方式では 57 台であったものが，並列化処理と Pulled Enumeration Operations を組み合わせた提案方式では 527 台と 9 倍以上もの台数を管理できるようになる．

上記より，従来方式では，ストレージ装置が 57 台以上，1 台のストレージ装置あたり 20,000Volume までのリソース数が限界であったが，提案方式により，約 9 倍の 527 台まで管理可能となる．これにより，約 5 年後には 100 台を越えるストレージ装置を所有すると予測される大型データセンタであっても，十分耐えうる情報収集方式が確立できた．

さらに，近年のストレージ管理ソフトウェアの動向としては，ストレージ装置を有効活用すべく，管理対象もストレージ装置だけでなく，仮想サーバやアプリケーションなど多種多様化してきている．そのため，ストレージ管理ソフトウェアは，ストレージ装置とサーバ間の構成を一元管理することが必要となる．また，本論文で述べたストレージ装置の台数増加に加え，仮想サーバの普及によりサーバ台数が増加している．

このような中，提案方式は，標準技術である CIM/WBEM に着目した方式のため，VMware や Xen といった CIM/WBEM をサポートしている仮想サーバや Windows に標準搭載されている WMI(Windows Management Instrumentation)のように，OS，アプリケーション，Hyper-V(仮想サーバ)を CIM でモデル化し構成情報を提供するようなものにも適用できる．そのため，ストレージ管理ソフトウェアにおける本提案方式の重要度は益々増すと考える．

8. おわりに

本論文では，ストレージ管理標準仕様の SMI-S を活用したストレージ管理ソフトウェア向けに，並列処理化と新オペレーションを組み合わせた構成情報収集方式を提案した．従来方式では，年々大規模化しているストレージ装置の台数やストレージ装置自身のリソース数の増加により，構成情報の収集時の処理時間と使用メモリ量がボトルネックとなり，一元管理が出来なくなってきている．本提案方式では，これらを改善したことで，従来方式の 9 倍以上ものストレージ装置の構成情報を一元管理出来るようになった．このように構成情報の収集処理のスケーラビリティを向上したことで，ストレージ管理ソフトウェアは，ディザスタリカバリやストレージ仮想化によって複数台のストレージ装置が連携し運用されるような大規模複雑なストレージ環境であっても，一元管理することが出来るようになった．

本論文では，構成情報収集モジュールに着目した．今後の課題は，収集した情報を格納する構成情報 DB および GUI/CLI といったストレージ管理ソフトウェアの他モジュールについても，大規模化するストレージ環境を見据えた設計を行うことである．

参考文献

1) IDC: Worldwide Enterprise Storage System 2009-2013 Forecast Update (2009)

2) SNIA: SMI-S.http://www.snia.org/forums/smi/tech_programs/smis_home/

3) SNIA: CTP Test Statistics. http://www.snia.org/forums/smi/tech_programs/ctp/generalinfo/statistics.html

4) DMTF: CIM. http://www.dmtf.org/standards/cim/

5) DMTF: WBEM. http://www.dmtf.org/standards/wbem/

6) 宮崎扶美, 兼田泰典, 篠原大輔, 藤田高弘, 古橋亮慈: ストレージシステム管理における CIM/WBEM 適用方式の研究，情報処理学会全国大会講演論文集, pp3.285-3.286 (2003)

7) Ramani Routray, Sandeep Gopisetty, Pallavi Galgali, Amit Modi, Shripad Nadgowda: “iSAN: Storage Area Network Management Modeling Simulation”, IEEE “Networking, Architecture, and Storage, 2007. NAS 2007. International Conference on”, pp199-208(2007)

8) 柴山司, 篠原大輔, 坂下幸徳, 小野卓也, 守島浩: SMI-S 準拠コピーサービスの実現方式, FIT2006 一般講演論文集, pp189-190 (2006)

9) (株)日立製作所: Hitachi Command Suite. http://www.hitachi.co.jp/products/it/storage-solutions/products/software/hsms/index.html

10) EMC: EMC Ionix ControlCenter. http://japan.emc.com/products/family/controlcenter-family.htm

11) IBM: IBM Tivoli Storage Productivity Center. http://www-06.ibm.com/systems/jp/storage/software/productivity/