

MOOS08

# Hadoop・HBaseを利用した J-PARC運転データアーカイブの現状

Status of J-PARC Operation Data Archiving Using Hadoop and HBase

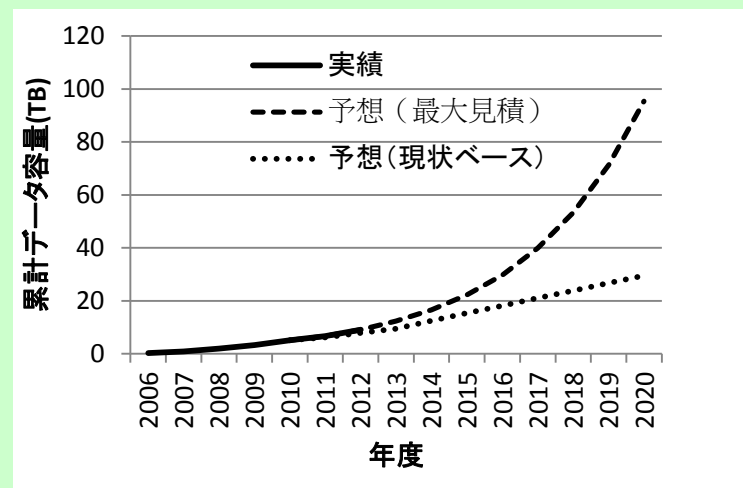
菊澤信宏 吉位明伸 池田浩 加藤裕子  
原子力機構／J-PARCセンター

1. 検討の背景
2. Hadoop・HBaseについて
3. システム構成
4. 可用性の向上とシステム管理
5. データ構造の検討
6. 現行システムとの性能比較
7. 課題
8. まとめ

現在J-PARCでは運転制御に関するEPICSレコードのデータをPostgreSQLを使用したデータベースシステムにアーカイブしている。

## <現行システムの課題>

- 将来的に100TB程度にまで増大する大容量のデータを一元的に管理することが困難。
- 性能やディスク容量の増強をシームレスに行う事が困難。
- 将来のハードウェアのライフサイクルエンドに伴う換装時のデータ移行が大変。



分散処理フレームワークHadoop  
分散データベースHBaseの利用を検討



Google社が自社のサイトやシステムで利用するために開発したファイルシステム(GFS)や分散処理機構(MapReduce)、分散データベース(BigTable)をモデルにオープンソースで開発されたもの

### <主な特徴>

- 数千ノードのクラスタ構成に対応し、ペタバイトクラスのデータのような超大規模処理にも対応
- スケーラブルなノード増強や交換への対応が可能

### <主要導入事例>

▪ Facebook    ▪ LINE    ▪ Ameba    ▪ Adobe    etc...

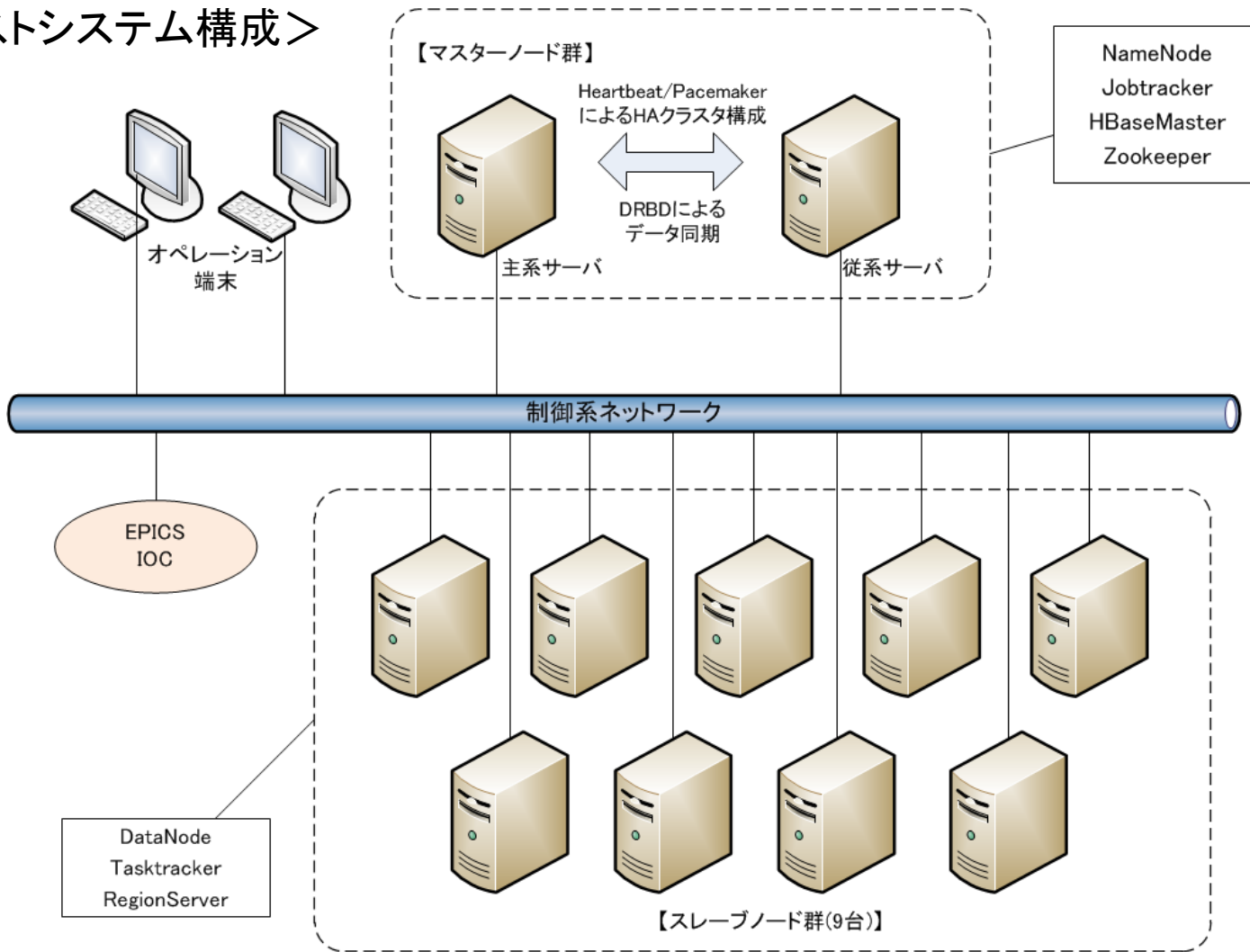
### <Hadoop・HBaseのバージョン遍歴概要>

	Hadoop			HBase	
2007	0.14.1				
2008				0.1.0	
2009	0.19.2	0.20.0		0.19.0	
2010				0.20.0	0.20.6
2011		0.23.0	1.0.0	0.90.0	
2012			1.0.4	0.92.0	0.94.0
2013		0.23.9	1.1.2	0.92.2	0.94.5
			1.2.0		0.94.10
			2.0.0		0.95.0
			2.0.5		0.95.1
future					..... 1.0.0?

近年、ビックデータを扱う必要性から非常に多くのNoSQLが乱立し、現在は過渡期。

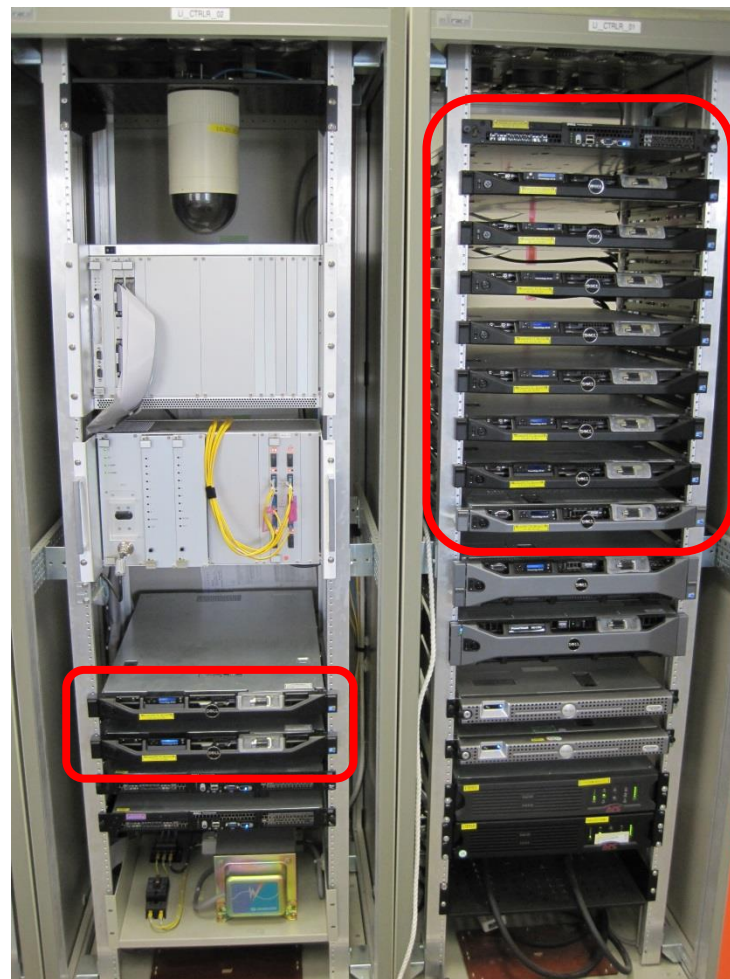
HBase	<ul style="list-style-type: none"><li>• HBaseはマスター・スレーブ型で強い一貫性が利点（最も新しいデータが見えることが保証される）</li><li>• 基盤となるHadoopのHDFSに単一障害点があるため、実運用では対策が必須</li></ul>
Cassandra	<ul style="list-style-type: none"><li>• CassandraはP2P型で可用性が利点（単一障害点がない）</li><li>• 一貫性と可用性のバランスを幾分調整可能</li><li>• ノードに障害が発生したときは、完全に回復するのに手作業が必要</li></ul>
Riak、Hibari	<ul style="list-style-type: none"><li>• 両者ともKey-Value型</li><li>• Erlang（歴史的にはJavaより古く、分散システムに特化し耐障害性に優れた言語）による実装にて高い品質が期待される</li><li>• Riakは可用性を重視、Hibariは一貫性を重視</li></ul>
その他、Javaで実装されたもの	<ul style="list-style-type: none"><li>• Voldemort: Key-Value型で可用性重視（P2P型）、Apache License 2.0</li><li>• okuyama: Key-Value型、GPLv3と商用のデュアルライセンス</li></ul>

## <テストシステム構成>



## <サーバ構成>

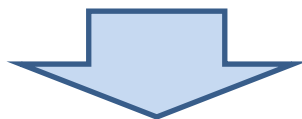
役割	構成
マスターノード (主系)	DELL PowerEdge R610 CPU: Intel Xeon E5620 (4Core 2.4GHz) MEM: 24GB (8GB x3,1333MHz,DDR3) HDD: 600GB SAS 10krpm x4 (RAID10)
マスターノード (従系)	DELL PowerEdge R200 CPU: Intel Xeon X3210(4Core 2.13GHz) MEM: 8GB (2GB x4,DDR2) HDD: 160GB SATA 7200rpm x1
スレーブノード	DELL PowerEdge R410 CPU: Intel Xeon E5620 (4Core 2.4GHz) <b>MEM: 24GB</b> (8GB x3,1333MHz,DDR3) <b>HDD: 2TB SAS 7200rpm x4</b> (RAID5)
ソフトウェア	OS: CentOS6.3 (Japanese) Java: JDK 1.6.0_45 Hadoop 1.0.4 HBase 0.94.5 Heartbeat 3.0.7 Pacemaker 1.0.12 DRBD 8.4.1 Ganglia 3.4.0



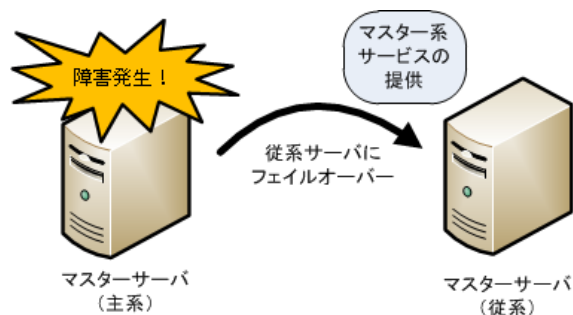


## <Hadoop・HBaseシステムの問題点>

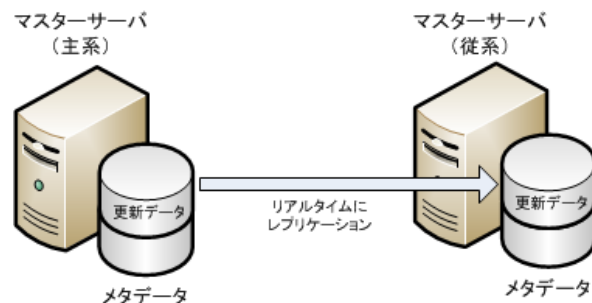
- マスターノードが単一障害点であり、マスターノードの停止がサービス停止に直結
- データ管理のメタデータが消失すると実質的にデータロス状態



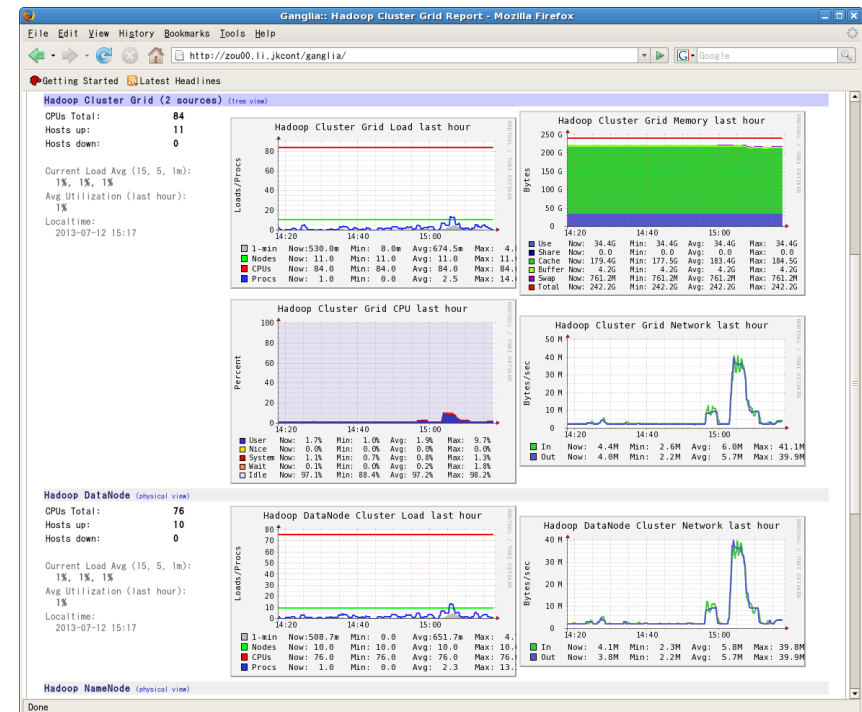
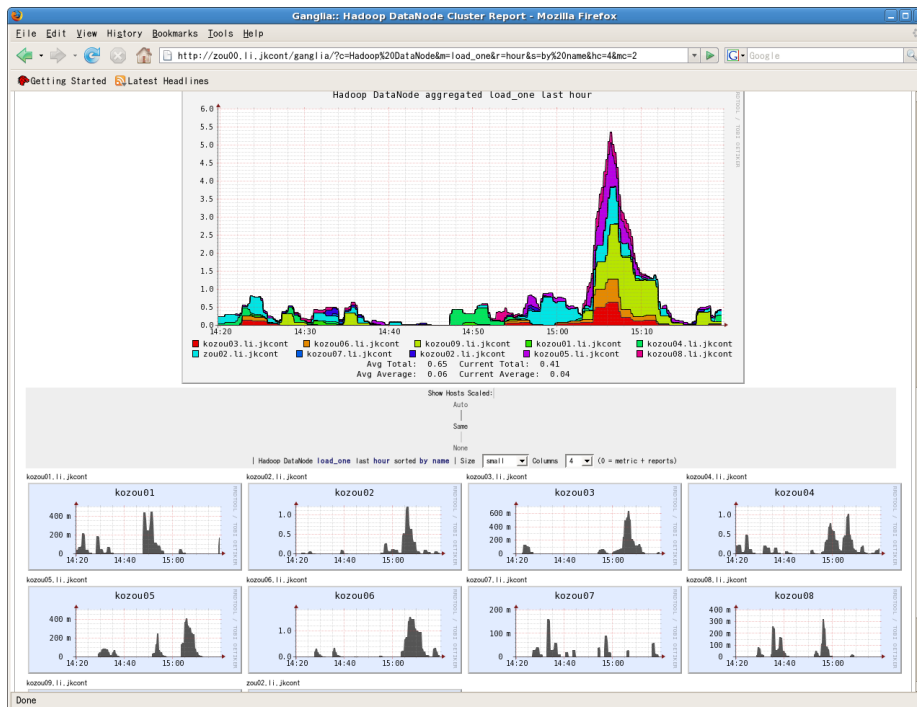
マスターノードはHeartbeat/Pacemaker  
でActive-StandbyのHAクラスタ構成に



メタデータ領域はDRBDで  
レプリケーション



## <Gangliaによるリソース監視>



各ノードのCPU、メモリ、ネットワーク帯域等のシステムリソース利用状況をリアルタイム且つ統合的に監視出来る

## < 現行システムのデータ構造概要 >

DataTable1\_YYYYMMDD

runtime	value				
timestamp1	Data_A1	Data_B1	Data_C1	Data_D1	Data_E1
timestamp2	Data_A2	Data_B2	Data_C2	Data_D2	Data_E2
timestamp3	Data_A3	Data_B3	Data_C3	Data_D3	Data_E3
...	...				

DataTable2\_YYYYMMDD

runtime	value		
timestamp1	Data_F1	Data_G1	Data_H1
timestamp2	Data_F2	Data_G2	Data_H2
timestamp3	Data_F3	Data_G3	Data_H3
...	...		

DataTable3\_YYYYMMDD

runtime	value			
timestamp1	Data_I1	Data_J1	...	Data_N1
timestamp2	Data_I2	Data_J2	...	Data_N2
timestamp3	Data_I3	Data_J3	...	Data_N3
...	...			

DataDefineTable

TableName	order	EPICSrecord
DataTable1	1	Record_A
DataTable1	2	Record_B
DataTable1	3	Record_C
DataTable1	4	Record_D
DataTable1	5	Record_E
DataTable2	1	Record_F
DataTable2	2	Record_G
DataTable2	3	Record_H
DataTable3	1	Record_I
DataTable3	2	Record_J
...	...	...
DataTable3	N	Record_N
DataTable4	1	Record_X
...	...	...

EPICSレコードと対応する格納  
テーブル名とデータ格納順序を  
示すテーブル

Valueカラム内に各データ(8byteのDouble型)を  
byte型の状態で繋げて格納

### <HBaseシステムにおけるデータ構造検討ポイント>

#### ■ 列志向型データベース

- 大量の行に対して少数の列でデータを集約する方が効率的 (key-value形式)

#### ■ 大きなサイズのテーブル保持が可能

- 1つのテーブルに多くのデータを格納させることが出来る

#### ■ row (所謂主キー) の昇順でデータを格納

- 単純増加するもの (Timestamp) をrowに割り当てると新規データ登録時に特定ノードに対して処理が集中

## <HBaseシステムにおけるデータ構造イメージ>

LINAC\_polling

row	value
LI_RECORD_MON_A-2013080513010000	Data_A1
LI_RECORD_MON_A-2013080513010100	Data_A2
LI_RECORD_MON_A-2013080513010200	Data_A3
...	...
LI_RECORD_MON_B-2013080513010000	Data_B1
LI_RECORD_MON_B-2013080513010100	Data_B2
LI_RECORD_MON_B-2013080513010200	Data_B3
...	...
LI_RECORD_MON_C-2013080513010000	Data_C1
LI_RECORD_MON_C-2013080513010100	Data_C2
LI_RECORD_MON_C-2013080513010200	Data_C3
...	...

LINAC\_event

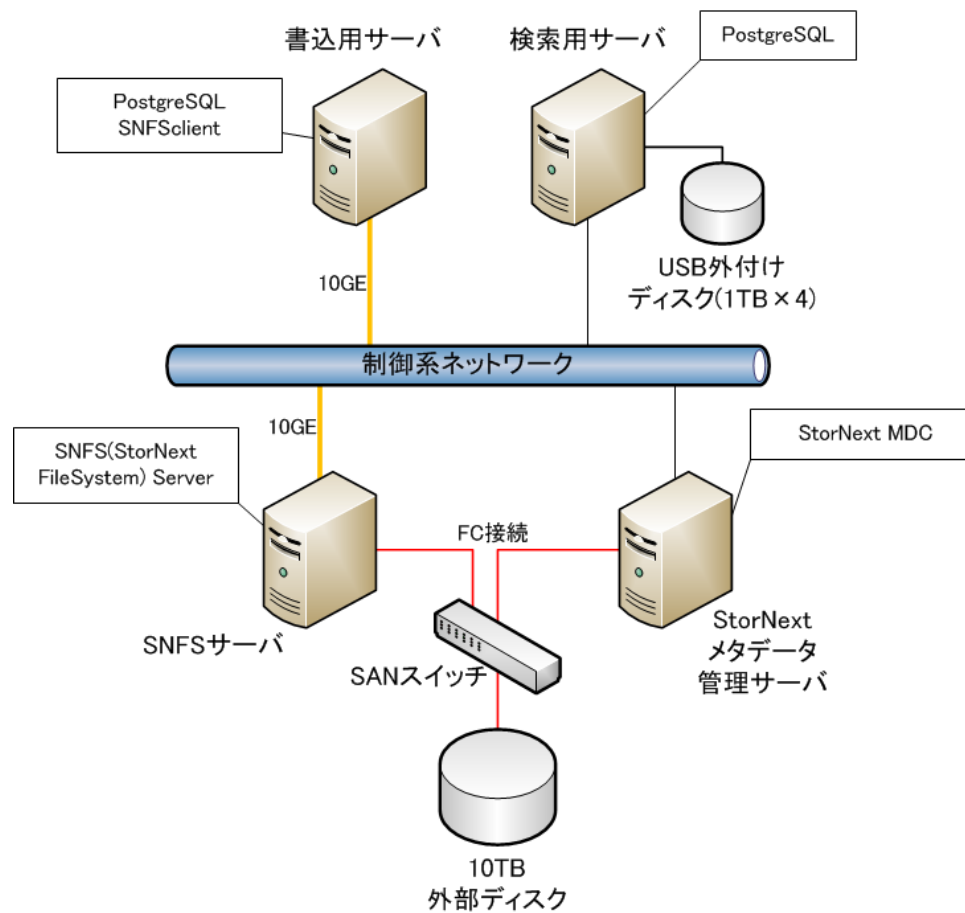
row	value
LI_RECORD_a-2013080513010000	Data_a1
LI_RECORD_a-20130805130107065	Data_a2
LI_RECORD_a-20130805130210186	Data_a3
...	...
LI_RECORD_b-20130805130023079	Data_b1
LI_RECORD_b-20130805130101099	Data_b2
LI_RECORD_b-20130805130209740	Data_b3
...	...
LI_RECORD_c-20130805130006507	Data_c1
LI_RECORD_c-20130805130101027	Data_c2
LI_RECORD_c-20130805130108135	Data_c3
...	...

特定のEPICS  
レコードの  
データが時  
系列順に並  
ぶ

新規データ  
は各EPICSレ  
コード時系  
列順の最後  
に挿入され  
る

## < 現行システム概要 >

役割	構成
データ書込用サーバ	DELL PowerEdge R200 CPU: Intel Xeon X3210 (4Core 2.13GHz) MEM: 8GB (2GB x4, DDR2) 内蔵 HDD: 160GB SATA ストア用HDD: 10TB (StorNextを用いた外部ファイルサーバのディスクを10GE経由でマウント)
データ検索用サーバ	DELL PowerEdge 860 CPU: Intel Xeon 3060 (2Core 2.4GHz) MEM: 4GB (2GB x2, DDR2) 内蔵 HDD: 250GB SATA ストア用HDD: 4TB (1TBx4、USB接続の外付けディスクを使用)



### <データ検索時間の比較>

任意の1種類のEPICSレコードのデータ5日分(1秒周期で約43万件)を取得するのに要した時間を測定。

システムとデータタイプ	検索時間(sec)
現行システム書込用サーバ(LLRF系データ)	50.5
現行システム検索用サーバ(LLRF系データ)	100.0
HBaseシステム(LLRF系データ)	20.9
現行システム書込用サーバ(イオン源系データ)	16.8
現行システム検索用サーバ(イオン源系データ)	26.9
HBaseシステム(イオン源系データ)	20.9

←数珠繋ぎデータの分離・抽出し処理の負荷が小さく、ディスクの高速性が優っている

- LLRF系データ:1846個のEPICSレコードが所属
- イオン源系データ:45個のEPICSレコードが所属

## ■システム可用性の向上

単一障害点のマスターノードのHAクラスタ構成にて可用性を向上



フェイルオーバーに5分程度の時間を要する為、クライアントアプリケーションがタイムアウトする恐れ



アプリ側でも必要なプロセスについて監視 & ダウン時の自動復旧を実装

## ■アプリケーション移植

HBaseシステムへのアクセスは基本的にJAVAのAPIを利用



Java以外の言語で作成されたアプリケーションについて再利用の制約が多い



Thrift、REST、Avroの様なゲートウェイAPIを通じたアクセスや、JNI (Java Native Interface)の利用可能性を検討。最終手段はJAVA化。



### ■ データバックアップ

スレーブノードのレプリカ保持、DRBDによるメタデータレプリケーションである程度のデータ保護性能を確保。



不適切な操作によるデータ消失や災害等によるノードの多数損壊等、データロストのリスクは存在



ディザスタリカバリサイトの構築、スナップショットの取得等の方策は多数存在するが、保護レベルやリカバリポイントにより多額のコストが必要。適正なレベルの対策を今後検討。

- Hadoop・HBaseを使用したテストシステムを構築し、実際にデータ収集・アーカイブを試験的に実施。十分実用に耐え得る性能や安定性を確認。
- 単一障害点となるマスターノードについてHAクラスタ化、メタデータのレプリケーションにより可用性とデータ保護性を向上。
- 今後はRCS、MRのデータも収集し、過去のデータを移行。対応アプリの開発を進めて本格運用を目指す。