Zabbix による SPring-8 制御ネットワーク監視系の構築

NETWORK MANAGEMENT SYSTEM FOR SPRING-8 ACCELERATOR CONTROLS

杉本 崇 *A)、石井 美保 A)、辻谷 健一 A)、上田 晃義 A)
Takashi Sugimoto* A), Miho Ishii A), Ken-ishi Tsujitani A), Teruaki Ueda A)
A)Japan Synchrotron Radiation Research Institute

Abstract

Local area network is used like a field bus at the SPring-8 accelerator controls. More than 300 network instruments are installed in the SPring-8 control system. To monitor all of network instruments for the SPring-8 control, we installed new monitoring system, Zabbix. By developing dedicate templates, we can monitor multi instruments on the one portal screen. We report system configuration and user interface of the new monitoring system.

1． 背景

SPring-8 は、独自開発された MADOCA フレームワークにより、ローカルエリアネットワーク (LAN) をフィールドバスとして使用する分散制御システムにより運転されている。SPring-8 の入射器および蓄積リングの機器制御のため、制御系 LAN は加速器施設のほぼ全域に敷設されている。制御系 LAN を構成するネットワーク機器数は 310 台に達しており、安定した利用運転のためにネットワーク機器の運用状況の監視が必要である。

我々のグループは SPring-8 制御系 LAN だけでなく、オフィスネットワーク (OA-LAN) などの SPring-8 共用施設のインターネット接続点から内部のネットワークを統一的に設計・構築・管理している。制御システムのセキュリティ担保の観点から、SPring-8 では OA-LAN と制御系 LAN の運用を明確に区別・隔離している。一方でネットワークシステム設計の共通化により、運用手順の標準化・省力化を進め、少人数による運用を可能としている。

SPring-8 のネットワークシステムでは、複数のメーカーの機材を意図して導入している。特定メーカー製品に共通した不具合・セキュリティホールによる全体への影響を避けるため、メーカーの撤退・廃業による調達継続性が失われることを避けるため、また、調達時の価格競争性を高めるためである。メーカー毎の設定方法の差違による工数増加を受け入れても、機材多様性のメリットが大きいと判断している。SPring-8 と SACLA 全体で 2016 年現在、ネットワークスイッチ 4 社、ファイアウォール 3 社、VPN 3 社、Wi-Fi 3 社の機材を併用している。(Table 1)

上記のように複数メーカーの機材を使用していると、運用・管理においても工数が増える。たとえば運用機材が Cisco 社のみであれば、Cisco Works, Cisco Prime Infrastructure などのメーカー純正の管理ソフトウェアにより一元管理することができる。一方で SPring-8 のように複数メーカーが混在する場合、汎用のネットワーク管理ソフトウェアを使用する必要がある。SPring-8 ではかつて各 LAN の監視システム整備のため、OpenView Network Node Manager や HP ProCurve Manager+を導入してきた。しかしながらその後、高価なライセンス費用の低減のため OpenView は運用を終了し、ProCurve Manager+も HP 社の事業方針の変更により 2015 年にメーカーサポートが終了している。

以上の状況から、新たにネットワーク監視システム構築する必要があった。近年、高機能ノードマネージャーがオープンソースで入手可能となっている。それらの中から、我々は Zabbix (`http://www.zabbix.com/`) による監視システムを構築した。Zabbix を選択したのは以下の利点からである。

- 監視方法の多様性（専用エージェント、スクリプト、SNMP polling/trap、Java JMX、IPMI）
- Low Level Discovery と OID Discovery 機能による柔軟なテンプレート
- トリガー機能によるイベント通知
- マップおよびスクリーン機能によるユーザーインターフェース

本稿では新監視システムの構成と、加速器オペレーターからみたユーザーインターフェースについて主に報告する。

Table 1: Network Instruments Which Used at SPring-8 and SACLA

	Ethernet Switch	Firewall	VPN	Wi-Fi
Cisco	✓	✓	✓	✓
DELL	✓	–	–	–
FortiNet	–	✓	–	–
Hitachi	✓	–	–	–
HP	✓	–	–	✓
Juniper	–	–	✓	–
PaloAlto	–	✓	–	–
YAMAHA	–	–	✓	✓

2． システム概要

Table 2 および Figure 1 に、SPring-8 における死活監視システムの計算機のスペックと構成を示す。Zabbix

* takashi.sugimoto@spring8.or.jp

Table 2: Hardware Specification of the Virtual Machine Environment, We Note That We Also Use Another Hardware for OA-LAN, Which Have the Same Specification

Lenovo System x3530 M4 SFF	
CPU	1× Xeon E5-2420v2, 6c12t, 2.2 GHz
Memory	2× 8 GBytes, DDR3-1600
Storage	2× 240 GBytes eMLC SSD (RAID 1)
Network	4× 1000BASE-T

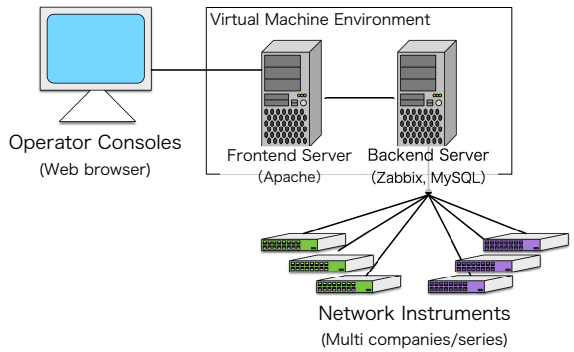

Figure 1: System components of the network management system. We assigned 2 and 8 threads to the frontend server and the backend server.

の基本コンポーネントは Web フロントエンド、サーバー、およびバックエンドデータベースで構成される。CPU リソースの分配のため実計算機上に 2 つの仮想計算機を構成し、フロントエンド部分とバックエンド部分に分割した。物理計算機および 2 台の仮想計算機の OS として Ubuntu Server 14.04 LTS を使用し、KVM と QEMU により仮想化環境を構築した。

仮想計算機のフロントエンド部は Web サーバー上で Zabbix フロントエンドの PHP スクリプトを動作させるため、CPU 2 thread、メモリ 1 GB、ストレージ 8 GB を割り当てた。バックエンド部は Zabbix サーバーと MySQL データベースを動作させるため、CPU 8 thread、メモリ 8 GB、ストレージ 160 GB を割り当てた。ストレージに SSD を採用することにより、高価なディスクアレイを使わずにデータベースに要求される I/O 性能を満たした。物理ネットワークは 1 ポートをオペレーターコンソールの Web ブラウザから通信するフロントエンド側に、1 ポートをネットワークスイッチと通信するバックエンド側セグメントに割り当てた。また、フロントエンド仮想計算機とサーバー・データベース仮想計算機間は、物理計算機内で仮想的に接続している。

3. ユーザーインターフェース

3.1 Web ブラウザによる操作

Figure 2 に監視システムのポータル画面を示す。ポータル画面では、1. Zabbix サーバーの動作状況、2. 全体マップ、3. ステータステーブルを表示する。Zabbix サーバーの動作状況では、表示情報の更新時間、登録ホスト数、収集データ点数、トリガー条件数を表示している。2016 年 7 月現在、302 ホスト、17 万点のデータ収集、および、1.1 万件のトリガー条件を監視している。全体マップでは、ネットワーク障害発生時のエリアが把握できる。各建屋・フロアに応じたサブマップ (Fig. 3) を定義し、サブマップ内の障害は上位の全体マップに障害発生アイコンとして表示される。障害を示すアイコンからドリルダウンすることによりサブマップに遷移し、障害発生機器の設置位置とホスト名、IP アドレスが表示される。ステータステーブルでは、障害発生エリアと機器メーカーを重篤度に応じて台数表示している。加速器オペレーターからネットワーク管理者への障害連絡時の補助情報となる。

ポータル画面から Graph をクリックし、対象ホストとインターフェースを選択することで収集データのグラフを表示できる。Figure 4 にグラフの例を示す。基幹ネットワークスイッチと SPring-8 中央制御室間の 1 ヶ月のトラフィックを表示しており、運転停止期間 (interval)、調整期間 (startup/study)、および、ユーザー運転期間 (operating) による通信量の傾向が確認できる。運転停止期間中も常時 250 Mbps の通信があり、ユーザー運転期間はさらに 50–80 Mbps の通信が上乗せされることが分かる。調整期間に通信量が上下しているのは、入射器から IP 伝送しているスクリーンモニタの利用によるものである。

3.2 メールによる障害通知

オペレーターコンソールの Web ブラウザ上だけでなく、ネットワーク管理者に対する 24 時間 365 日の通知のためにメールが自動送信される。過剰な通知を避けるため、トリガーの重要度 (severity)、事前のメンテナンス期間設定に応じて発報すべき条件を定義した。

4. 今後の計画

2014 年度に Pring-8 キャンパスの OA-LAN、次いで 2015 年度より SPring-8 制御系 LAN に Zabbix によるネットワーク監視システムを導入してきた。2016 年夏より SACLA 制御系 LAN に導入するため、システムを構築中である。今後、SPring-8/SACLA の両実験・データ収集系 LAN、および、データ解析系 LAN にも監視システムを順次整備する計画である。Table 3 は、SPring-8 で運用している機器とテンプレートのリストである。未作成のものについても今後、Enterprise MIB を活用したメーカー別テンプレートを製作予定である。

5. 謝辞

MySQL データベースのチューニングについて、理化学研究所エンジニアリングチームの丸山氏より助言をいただいたことをここに感謝する。

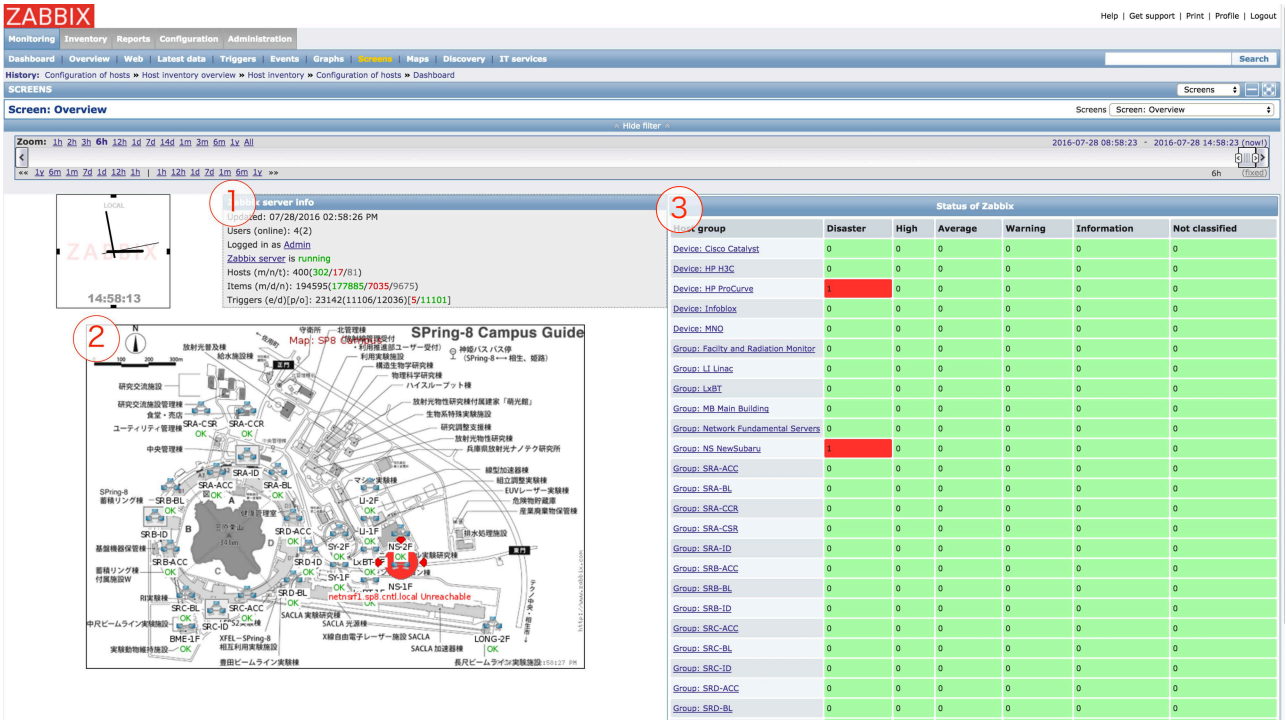

Figure 2: Portal screen of the network management system. 1. Running status of the Zabbix server. 2. Map view of network status. Each icon represents submap of buildings/floors. Network trouble occurred withing the submap of red icons. 3. Table view of network status. The red cell represents network trouble, which sorted by area and by device manufacturers.

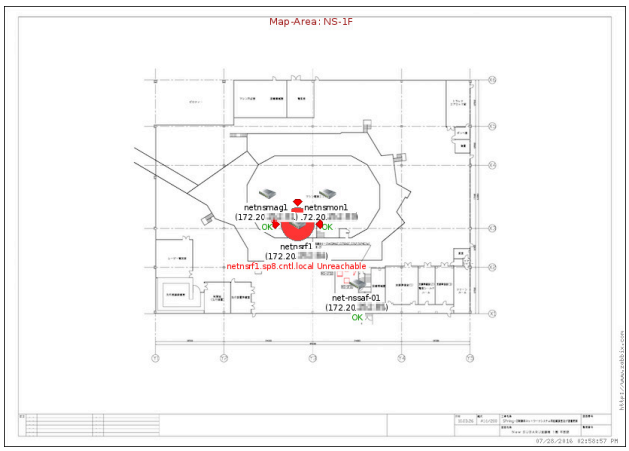

Figure 3: Example of the floor map. Operator can identify location of the troubled instruments. Hostname and IP address are also shown near the trouble icon.

6． (付録) システムチューニング

本文中に記したように、SPring-8 制御系 LAN の監視システムは 10 万点を超えるデータ点と 1 万点を超えるトリガー条件を収集・監視しており、パラメーターチューニングが必要であった。参考として、SPring-8 の監視システムにおいて特にチューニングした点を記す。

6.1 Zabbix サーバー

以下は `/etc/zabbix/zabbix_server.conf` においてデフォルトから変更したパラメーターである。

```
StartPollers=64              # default 5
StartPingers=8               # default 1
CacheSize=2G                 # default 8M
HistoryCacheSize=32M         # default 8M
TrendCacheSize=128M          # default 4M
HistoryTextCacheSize=32M     # default 16M
ValueCacheSize=128M          # default 8M
Timeout=10                   # default 3 (秒)
```

StartPollers (Poller プロセス数) 監視データ点が多いため、標準の polling プロセス数では指定した間隔で終了せずに遅延していた。polling プロセスの並列度を上げることで分散処理を図った。並列プロセス数は CPU で使用可能な thread 数を上限にするのが一般的だが、polling 処理の大部分は応答待ちであるため thread 数を増やしても効果が得られた。

StartPingers (Ping プロセス数) polling プロセス数と同様に ping プロセスが指定した間隔で終了しないため、ping プロセスの並列度を上げて分散処理を図った。

CacheSize (設定情報キャッシュサイズ) Low Level Discovery の活用により、監視データ点が多く設定情報が巨大となっている。設定情報を都度データベースにクエリさせないため、設定情報キャッシュサイズを

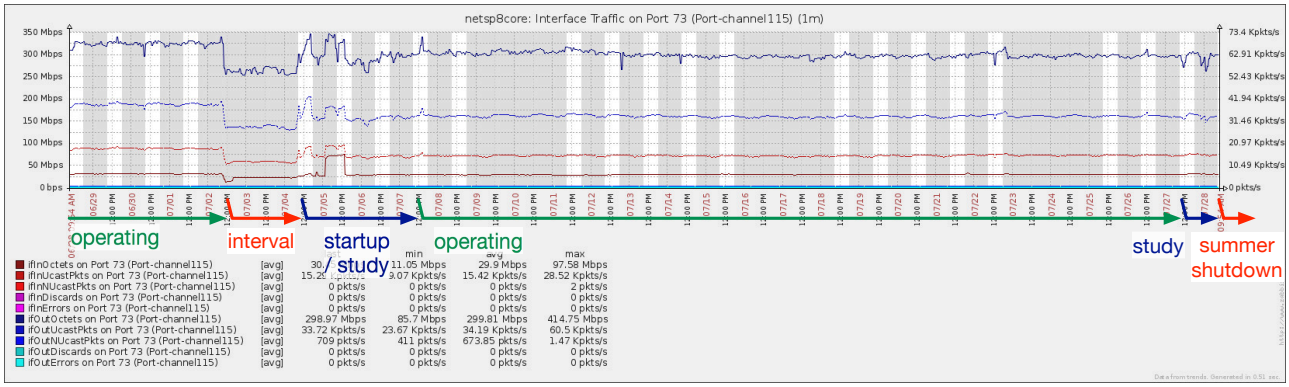

Figure 4: Network traffic trends between core network switch and central control room in July, 2016.

Table 3: Supported Zabbix Templates, The Checkmarks (✓) Are in Service on the Monitoring System, and the Others Are to Be Supported

Company	Series	Supported
Cisco	Catalyst	✓
	Nexus	–
	ASA	✓
	WLC	✓
DELL	Force 10	–
FortiNet	FortiGate	✓
Hitachi	Apresia Switch	✓
	Apresia Light	–
HP	ProCurve	✓
	H3C	✓
	Colubris/MSM	–
Juniper	SRX	✓
	SA/MAG	✓
PaloAlto	PA Firewall	✓
YAMAHA	RTX	–
	WLX	–

増やした。パラメーター値は Zabbix エージェントで取得できる “configuration cache % free” の値が 75%を下回らないよう決定した。

*ValueCacheSize (*値キャッシュサイズ*)* トリガーの条件判断時の値をオンメモリにキャッシュし、データベースへのクエリに伴うパフォーマンスの低下を減らした。パラメーター値は Zabbix エージェントで取得できる “value cache effectiveness” の値が 90%を下回らないよう決定した。

Timeout (SNMP 応答タイムアウト*)* SPring-8 制御系ネットワークには CPU 性能が低い古いネットワーク機器が含まれているため、SNMP polling の応答が遅くタイムアウトに達する場合があった。Timeout パラメーターを延ばすことでタイムアウトが発生しなくなった。

その他のキャッシュパラメーター Zabbix エージェントで取得できる各 “cache % free” 値が 90%を下回らないよう決定した。

6.2 MySQL サーバー

多数のデータ点を収集しているため、データベースへの負荷も高い。内部データベースエンジンは InnoDB を使用し、データベースのチューニングを実施した。

以下は `/etc/mysql/my.cnf` においてデフォルトから変更したパラメーターである。

```
[mysqld]
 :
thread_cache_size=400     # default 8
max_connections=400       # default 100
```

*max_connections (*最大同時接続数*)* Zabbix において poller, pinger プロセス数を増やしたため、データベースへの同時接続数も増えている。同時接続数を増やすことで、データベース側の接続数超過エラーの発生を抑制した。パラメーター値は `show variables like "%max_connections%"` コマンドによる最大同時接続数の履歴に対して約 3 倍の余裕をみて決定した。

*thread_cache_size (*切断後も維持するスレッド数*)* thread_cache_size パラメーターは、最大同時接続数と同等にすることが望ましいため、max_connections と同じ値に決定した。

6.3 テンプレートとトリガー条件

Zabbix のトリガー条件の設定は柔軟性が高いため、最新値 (`.last()` 関数) と前回値 (`.prev()` 関数) を容易に取得し、比較できる。しかしながら、SQL における前回値の取得は、最新値の取得と比べて負荷が高い。当初作成したテンプレートでは、Low Level Discovery で定義される個々のネットワークインターフェース等に対して up→down などの状態遷移を全て監視していたため、データベースへの負荷が高く安定運用できていなかった。システム全体の負荷を精査し、Low Level Discovery に含まれるトリガーは状態遷移を使用しないことで、安定動作を実現できた。