SuperKEKB への CSS V4 アラームシステム導入における性能評価

EVALUATION OF THE CSS V4 ALARM SYSTEM FOR SUPERKEKB

廣瀬雅哉#, A), 中村達郎 B), 佐々木信哉 B), 中村卓也 C), 浅野和哉 A)
Masaya Hirose #, A), Tatsuro Nakamura B), Shinya Sasaki B), Takuya Nakamura C), Kazuya Asano A)
A) Kanto Information Service Co.,Ltd.
B) KEK
C) Mitsubishi Electric System & Service Co.,Ltd.

Abstract

CSS version 3 alarm system was operated stably during phase 1 operation of SuperKEKB in the most cases. However, the alarm system has a problem related to connection with the EPICS records. To resolve the problem, we evaluated CSS version 4 alarm system which is current development major version. The test of connection to EPICS records revealed that connection become more stable and faster in CSS version 4.4 because of using updated CA library, CAJ 1.1.15. We also take place a performance test of CSS version 4.4 alarm system, and conclude that the system has enough performance. In this paper, we report results of the evaluation test and the performance test.

1. はじめに

SuperKEKB では、KEKB での電子・陽電子衝突頻度を約 40 倍に高めることを目指しており、このような高輝度加速器の運転下において、安定して動作可能なアラームシステムの構築が重要である。

SuperKEKB 加速器では、2016 年 2 月から 2016 年 6 月に Phase-1 運転が行われた。Phase-1 では、Control System Studio（CSS）[1][2]のメジャーバージョン 3（以下 V3 のようにする）のアラームシステムを整備し、運用した[3][4]。このアラームシステムは概ね順調に動作していたが、Experimental Physics and Industrial Control System（EPICS）[5]のレコード（以下レコードという）との接続に関するトラブルがしばしば発生した。より安定したアラームシステムを実現するためには、より安定したレコードとの接続が必要となる。

CSS の開発プロジェクトでは、メジャーバージョン 4（以下 V4 のようにする）の開発が日々進められている。V4 では、CSS 内部でレコードを扱う部分（PV layer）が変更されている他、Channel Access（CA）ライブラリがバージョンアップしており、より安定したレコードとの接続が期待される。

そこで我々は、SuperKEKB への V4 アラームシステム導入を検討するため、その動作試験及び性能評価を行った。本稿では、この詳細について報告する。

2. CSS アラームシステムの概要

CSS アラームシステムは、Figure 1 のように複数のソフトウェアから構成される。このシステムでは Alarm Server が中心となり、Java Message Service（JMS）を介して各種ソフトウェアとメッセージ通信を行う。

Alarm Server はリレーショナルデータベース（RDBMS）からアラーム構成情報を読み込み、レコードのアラームを監視する。アラームが発生した場合は、Alarm Server が JMS を経てグラフィカルユーザインタフェース（GUI）やアナウンスツールなどに通知する他、JMS2RDB がアラームメッセージを RDBMS に記録する。記録したアラームメッセージは、CSS が提供する Client GUI で閲覧することができる。これにより、ある期間のアラーム発生状況の確認や、アラームの発生頻度が高いレコードの確認などを後から行うことができる。

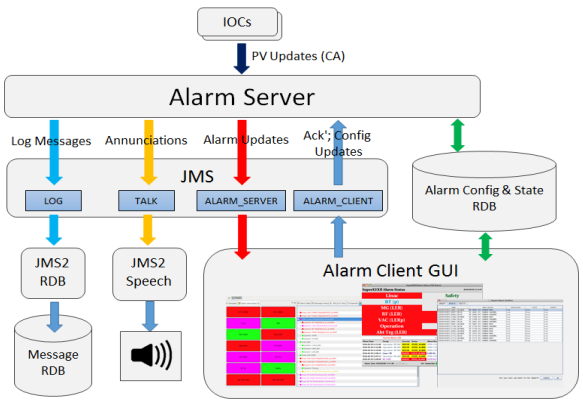


Figure 1: Overview of CSS alarm system.

3. Phase-1 における CSS アラームシステムの運用状況

3.1 システム構成

Phase-1 運転では、バージョン 3.1.2（以下 V3.1.2 のようにする）の CSS アラームシステムを整備し、運用した。このとき、V3.2.16 の CSS アラームシステムも整備し、試験的に運用した。RDBMS には PostgreSQL[6]、JMS には Apache ActiveMQ[7]を採用し、これらをアラームシステム用の 1U サーバ上に整備した。オペレータやユーザなどが利用するクライアントツールには、CSS が提供する Client GUI に加え、独自に開発した GUI も利用した[4]。

kan-hiro@post.kek.jp

3.2 運用状況

2 つのバージョンとも概ね順調に動作していたが、いくつか問題が発生した。その中でも重要な問題が 2 つあった。一つはアラームが発生した場合、発生時のレコードの値を文字列形式で RDBMS に記録するようになっているが、この文字列が 100 文字を超えると記録されない問題である。これは V3.1.2 のみで発生した問題であり、V3.2.16 では既に修正されていることが確認されているため、バージョンアップすることにより解決できる。もう一つは Input Output Controller（IOC）や CA-Gateway の再起動が行われた際にレコードが再接続しない問題である。この問題はどちらのバージョンでもしばしば発生しており、システム管理者が Alarm Server を再起動するなど、状況に応じて対処していた。

4. レコードとの接続に関する動作試験

CSS アラームシステムでは、レコードとの接続に関する部分はサードパーティーの CA ライブラリが担っている。CA ライブラリとして、CAJ[8]と JCA[9]が用意されており、デフォルトでは CAJ が使われる。ここで、CSS のバージョン毎の PV layer および CA ライブラリのバージョンを Table 1 に示す。

4.1 概要

Table 1 に示す通り、CSS のバージョンが上がるにつれ、CAJ のバージョンも上がっている。そこで我々は Phase-1 運転で発生した問題の再現性についても確認するため、使用している CAJ のバージョンが異なる V3.2.16、V4.3.3、V4.4.1 の 3 つのバージョンにおいて、レコードとの接続に関する動作試験を実施した。

試験環境の構成を Figure 2 に示す。Alarm Server や JMS、RDBMS を動作させるための 1U サーバと、IOC を動作させるための PC を 2 台用意し、これら 3 台が同じネットワークに接続される構成とした。OS は CentOS7.1 に統一し、RDBMS には PostgreSQL9.5、JMS には Apache ActiveMQ 5.11.1 を採用した。

IOC のレコード数は 3,000 とし、この IOC をそれぞれの PC で動作させるよう整備した（レコード総数 6,000）。IOC の起動・停止ログは、Zero Message Queue（ZMQ）[10]を利用して、一箇所で管理するように整備した。

なお、レコードとの接続に関する動作を確認するため、以下のように実施した。

i. 2 つの IOC を起動する。
ii. Alarm Server を起動し、全レコードと接続完了後、2 つの IOC を停止する。
iii. 2 つの IOC を停止してから一定時間（30 秒または 10 分）経過後に、1 つもしくは 2 つの IOC を起動する。
iv. IOC を起動した後、IOC-Shell に組み込まれている casr コマンドを 10 秒毎に実行して接続状況を確認する。

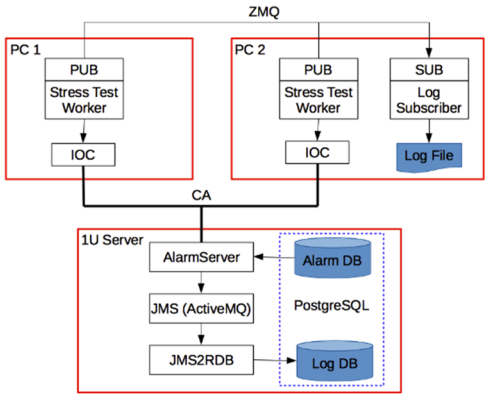


Figure 2: Overview of test of connection to EPICS records of CSS alarm system.

4.2 試験結果

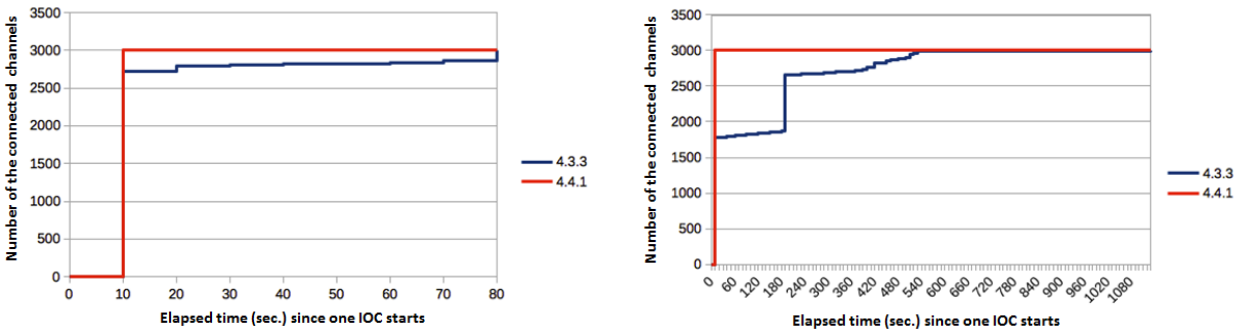
V4.3.3 と V4.4.1 における全レコードが接続完了となるまでの接続状況の違いを Figure 3 および Figure 4 に示す。Figure 3 は Alarm Server と接続していた 2 つの IOC を停止後、片方の IOC のみを再起動した場合、Figure 4 は両方の IOC を再起動した場合の接続状況を示している。なお、V3.2.16 と V4.3.3 で動作に大きな違いがなかったため、V3.2.16 の図は省略する。V4.4.1 では、どの場合においても約 10 秒で全レコードとの接続が完了した。しかし、V3.2.16 および V4.3.3 では、全レコードが接続完了となるまでにかかる時間が、IOC を停止してから起動するまでの時間や接続状況によって違うことが分かった。両方の IOC を停止してから片方の IOC のみを起動した場合は、30 秒後だと約 80 秒、10 分後だと約 1140 秒であった。しかし、両方の IOC を再起動した場合は、いずれも約 20 秒であった。また、V3.2.16 と V4.3.3 では全レコードに再接続できなくなってしまう問題も発生した。今回の試験では、V4.3.3 以前では以下の状況の場合にレコードとの接続が遅くなる可能性があることが分かった。

- Disconnect 状態のレコードの絶対数が多い。
- CA サーチの成功率が悪い。
- Beacon Anomaly が正常に届いていない。

今回の試験では、V4.3.3 と V4.4.1 でレコードとの接続の動作に違いがあることが分かった。そこで我々は、これらのバージョンで使われている CAJ 1.1.14 と CAJ 1.1.15 の CA サーチの実装の違いについて調査を行った。

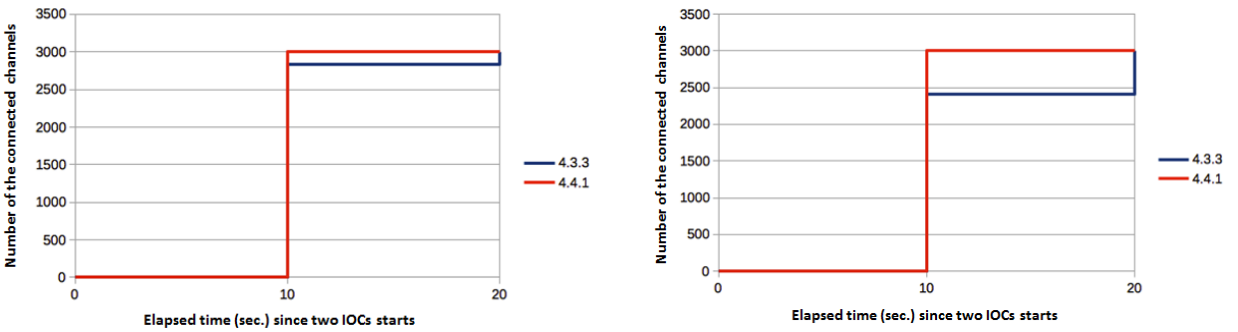
Table 1: Version of PV Layer and CA Libraries for Each CSS Version

CSS version	3.1.x	3.2.x	3.3.x	4.0.x ~ 4.3.x	4.4.x
PV layer	utility.pv	pvmanager	pvmanager	vtype.pv	vtype.pv
CAJ version	1.1.10	1.1.13	1.1.14	1.1.14	1.1.15
JCA version	2.3.6	2.3.6	2.3.6	2.3.6	2.3.6



(a) After 30 seconds　　(b) After 10 minutes

Figure 3: These are the connection state when only one IOC is started after a certain period of time has elapsed since stopping all the IOCs.



(a) After 30 seconds　　(b) After 10 minutes

Figure 4: These are the connection state when all IOCs started after a certain period of time has elapsed since stopping all the IOCs.

4.3　CAJ 1.1.14 における CA サーチの実装

CA プロトコルでは、サーチリクエストという UDP フレームを指定された CA アドレスリストに対して送信してレコードを検索し、接続を試みる CA サーチが行われる。1 フレームでサーチ出来るレコード数は決まっているため、全レコードが Connect 状態となるまでサーチリクエストの作成と送信が繰り返される。サーチリクエストの作成と送信は Timer Task（以下タスクという）によって行われ、32 ミリ秒から 32 秒までの実行周期が異なる 11 個のタスクがデフォルトで用意される。Disconnect 状態のレコードはタスクの First-in First-out（FIFO）バッファに追加され、タスク実行時に FIFO からレコードを取り出してサーチリクエストを作成し、送信する。なお、タスクはタスクを実行する実行タイマーの 2 分ヒープによる優先度付きキューで管理・実行されるようになっている。サーチリクエストを送信したが、レスポンスが無かったレコードは次の遅い周期のタイマーの FIFO に追加される。そのため、しばらくレスポンスが無いレコードは最長周期のタスクの FIFO に追加される。ただし、以下の場合には Disconnect 状態のレコードを早い周期のタイマーに移動させ、サーチリクエストが多く飛ぶように実装されている。

- サーチリクエストの成功率が高い場合。
- Beacon Anomaly が発生した場合。

また、サーチリクエストが飛ばなくなった原因は、タスクの実行アルゴリズムに問題があると考えている。CAJ 1.1.14 では、無限ループの中で 2 分ヒープのルートのタスクをキューから取り出して実行を繰り返すような実装がされている。このキューにタスクが追加されるケースは、以下の通りである。

- FIFO が空のタスクに新しくサーチするレコードを追加した時
- タスク実行終了時（サーチ終了時）に FIFO にレコードが残っている場合
- サーチしたすべてのレコードからレスポンスがあった時

追加されたタスクは実行終了後に削除され、FIFO にレコードが残っている場合は、実行タイマーの 2 分ヒープに再び追加される。しかし、今回の試験では 1 秒周期のタスクがある時から抜け落ちてしまい、実行されていないような状態となっていた。抜け落ちたタスクは FIFO にレコードを持っているため、2 分ヒープに再び追加されることが出来なくなった。このタスクに全てのレコードが追加され、サーチされることなく保持され続けていた。

4.4 CAJ 1.1.15 における CA サーチの実装

CAJ 1.1.15 では、CA サーチ部分が再実装されてシンプルになっている。まず、タスクは Disconnect 状態のレコードの数だけ作成される。タスクが実行されると、保持しているレコードをサーチリクエストに追加する。サーチリクエストは、以下のときに送信されるようになっている。

- フレームが満杯で、レコードを追加できないとき。
- レコードを追加するタスクが無いとき。

タイマー周期は、まず 3 ミリ秒に設定されてから徐々に長くなるように設定される。この周期は、フレームにレコードを追加したときに更新される。サーチリクエストを送信し、レスポンスのあったレコードのタイマーは削除される。なお、Beacon Anomaly があった場合は、すべてのタイマーの周期を 0 秒に戻すよう実装されている。

また、タスクは CAJ 1.1.14 と同様にタスクを実行する実行タイマーの 2 分ヒープによる優先度付きキューで管理・実行されるようになっている。そのため、タスクが抜け落ちてしまう可能性も考えられる。しかし、一つのタスクで 1 レコードしか持っていないため、仮に抜け落ちたとしてもその影響は低いと考えている。

5. パフォーマンス測定

我々は、バージョンを上げることによってパフォーマンスに影響があるかを確認するため、V4.4.1 アラームシステムのパフォーマンス測定を行った。V4.4.1 では CAJ 1.1.15 を利用しているため、ネットワークトラフィック量（以下トラフィック量とする）が多くなることが予想された。そこで、V4.3.3 で発生するトラフィック量も測定し、比較した。

5.1 概要

測定環境は Phase-1 運転前に行われた試験[3]を参考に構築した。測定環境の構成図を Figure 5 に示す。Alarm Server 等を動作させる計算機の基本情報は Table 2 に示す通りである。IOC は、アラームを監視するレコードを 50,000 件とし、40 秒おきに 200 件のレコードのアラームが変化するようにした。測定中は IOC を任意のタイミングで起動・停止させた。このときの CPU 使用率やトラフィック量などは dstat コマンドを用いて監視した。

5.2 測定結果

測定結果を Figure 6 および Figure 7 に示す。Figure 6 は V4.4.1 の CPU 使用率を示している。IOC の起動・停止直後などは一時的に高くなるが、その場合でも 60 %程度であった。Figure 7 は V4.3.3 と V4.4.1 のトラフィック量を示している。IOC が停止しているときのトラフィック量は V4.4.1 のほうが多くなっているが、1 MB/s 以下であるため、運用上大きな影響は無い。なお、トラフィック量が 10 MB/s を超える時があったが、このほとんどは CSS Client GUI との通信によるものであった。

今回の測定では、バージョンを上げてもパフォーマンスに影響はないと考えている。

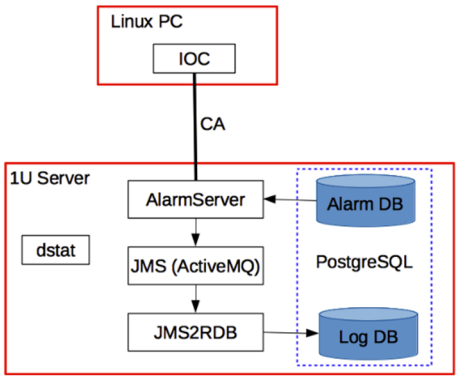


Figure 5: Overview of performance test of CSS alarm system.

Table 2: Specification of Server Computer

OS	CentOS 7.1
CPU	Intel Xeon E5-2603 1.6GHz
Memory	32GB
Disk Type	SSD

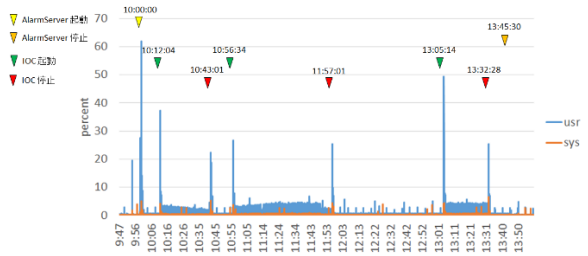
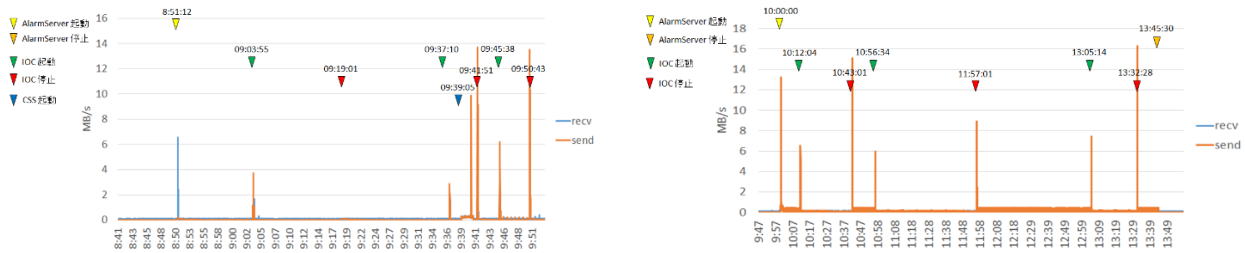


Figure 6: CPU usage rate for version 4.4.1 when IOC was started and stopped.



(a) Version 4.3.3

(b) Version 4.4.1

Figure 7: Comparison of network traffic volume that occurred when IOC was started and stopped.

6. まとめ

我々は SuperKEKB への CSS V4 アラームシステム導入を検討するために、レコードとの接続に関する動作試験およびパフォーマンス測定を行った。

レコードとの接続に関する動作試験では、更新された CA ライブラリである CAJ 1.1.15 を使用している V4.4 がより安定かつ早いことが分かった。また、V4.4 のパフォーマンス測定も実施し、SuperKEKB のアラームシステムとして導入するには十分なパフォーマンスであることも確認した。

今後は Phase-2 運転に向け、より安定したアラームシステムを実現するために、V4.4 以降のアラームシステムの整備を進めていきたい。

参考文献

[1] Control System Studio;
http://controlsystemstudio.org
[2] Control System Studio for KEK;
http://www-linac.kek.jp/cont/epics/css/
[3] T.Nakamura *et al*., “SuperKEKB への CSS アラームシステム導入における性能評価”, Proceedings of the 12th Annual Meeting of Particle Accelerator Society of Japan, Fukui, August. 5-7, 2015;
http://www.pasj.jp/web_publish/pasj2015/proceedings/PDF/WEP1/WEP110.pdf
[4] T.Nakamura *et al.*, “SuperKEKB の CSS アラームシステム運用状況”, Proceedings of the 13th Annual Meeting of Particle Accelerator Society of Japan, Chiba, August 8-10, 2016;
http://www.pasj.jp/web_publish/pasj2016/proceedings/PDF/TUP0/TUP095.pdf
[5] Experimental Physics and Industrial Control System;
http://www.aps.anl.gov/epics/
[6] PostgreSQL;
https://www.postgresql.org
[7] Apache ActiveMQ;
http://activemq.apache.org
[8] Channel Access for Java;
http://epics-jca.sourceforge.net/caj/index.html
[9] Java Channel Access;
http://epics-jca.sourceforge.net/jca/index.html
[10] ZeroMQ;
https://en.wikipedia.org/wiki/ZeroMQ