Schematy monitoringu

Wstęp

Schemat monitorujący określa zarówno transformacje przeprowadzane na danych przesłanych przez skrypt monitorujący, jak i graficzną reprezentację tych danych. Z tego powodu, każdy schemat monitorujący (oprócz ogólnego) jest ściśle związany ze skryptem, którego zadaniem jest przesłać aktualne dane o stanie zadania w odpowiednim formacie. Skrypty monitorujące dla zdefiniowanych schematów są dostępne na wszystkich maszynach w infrastrukturze PL-Grid. Poniżej prezentujemy aktualnie dostępną listę schematów monitorowania.

Schematy danych

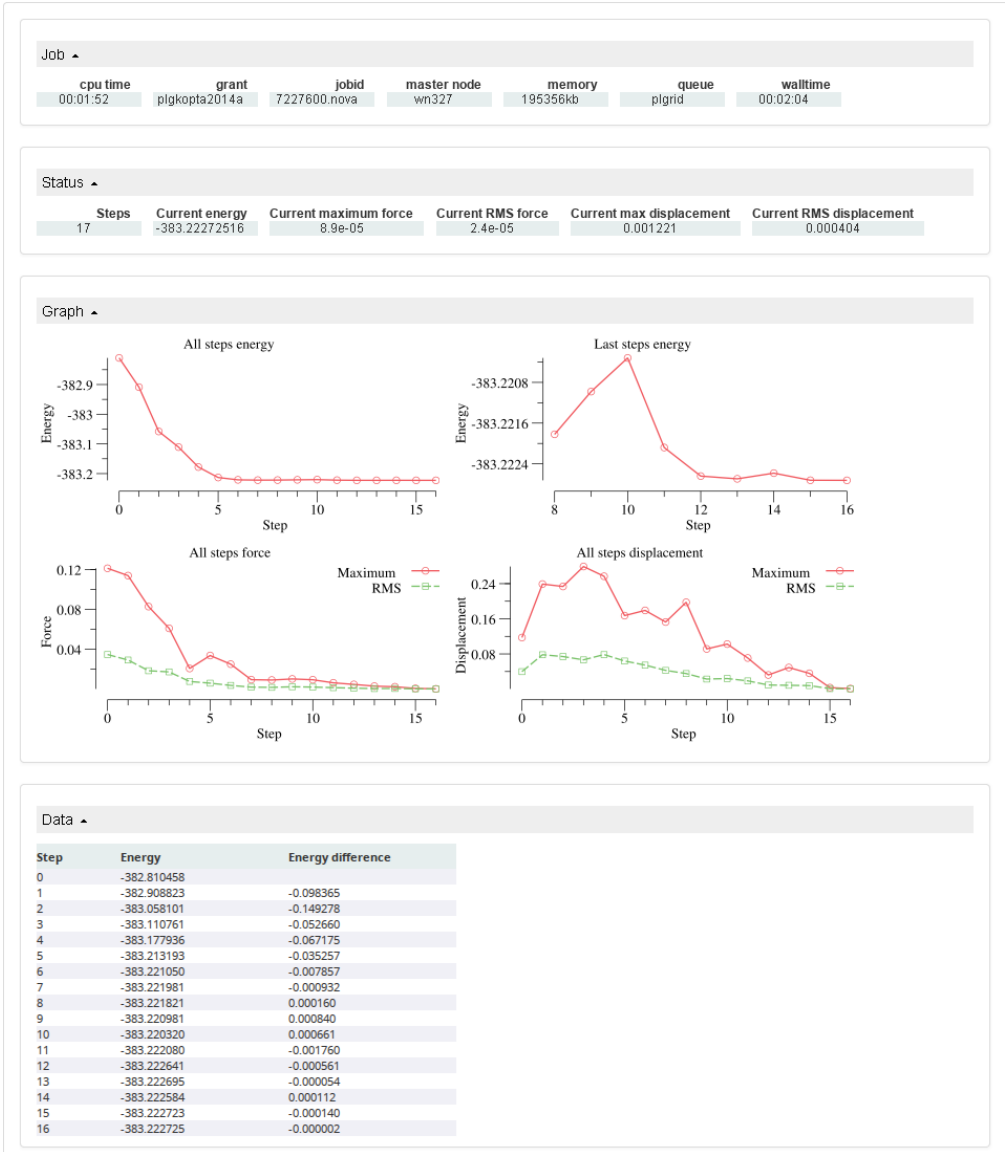
generic

Najbardziej ogólny schemat służący do prezentowania danych w formacie tekstowym. Jeśli użytkownik nie zdefniuje własnego skryptu monitorującego, używany jest domyślny, który wysyła ostatnich 10 linii standardowego wyjścia aplikacji. W przypadku własnego skryptu monitorującego, użytkownik musi skopiować go do katalogu roboczego (np. za pomocą dyrektywy *stage-in-file*). Szczegóły dotyczące implementacji skryptu monitorującego znajdują się poniżej.

gaussian

Schemat przeznaczony do ogólnego monitorowania eksperymentów uruchamianych za pomocą aplikacji Gaussian. Pośród prezentowanych danych znajdują się:

- standardowa sekcja *Job* z parametrami środowiska uruchominiowego (zużycie procesora, nazwa grantu, lokalny identyfikator zadania, nazwa wezła, zadeklarowana pamięć, nazwa kolejki oraz czas trwania aplikacji),
- sekcja Status zawierające informacje o liczbie wykonanych kroków symulacji, aktualnej energii, parametrach 'MAX Force', 'RMS Force', 'MAX Displacement' oraz 'RMS Displacement',
- sekcja Graph zawiera wykresy prezentujące przebieg parametrów sekcji Status w czasie trwania całej symulacji,
- sekcja Data prezentuje dokładne wartości energii we wszystkich dotychczas obliczonych krokach czasowych.



Istnieje możliwość dodania informacji tekstowych do prezentowanych danych. W tym celu, użytkownik musi utowrzyć własny skrypt powłoki który zostanie skopiowany do katalogu roboczego oraz wskaże na niego w zmiennej środowiskowej QCG_MONITOR_USER_SCRIPT (za pomocą dyrektywy environment). Skrypt ten powinien wypisywać dane w formacie opisanym w punkcie Skrypty monitorujące.

Poniżej umieszczony jest przykładowy opis zadania, który korzysta z dodatkowych informacji tekstowych.

```
#QCG output=output
#QCG stage-in-file=Naphthalene.gjf
#QCG stage-in-file=nfock.sh

#QCG application=g09
#QCG argument=Naphthalene.gjf

#QCG monitor
#QCG environment=QCG_MONITOR_USER_SCRIPT->nfock.sh
```

Plik wejściowy dla zadania dostępny jest tutaj.

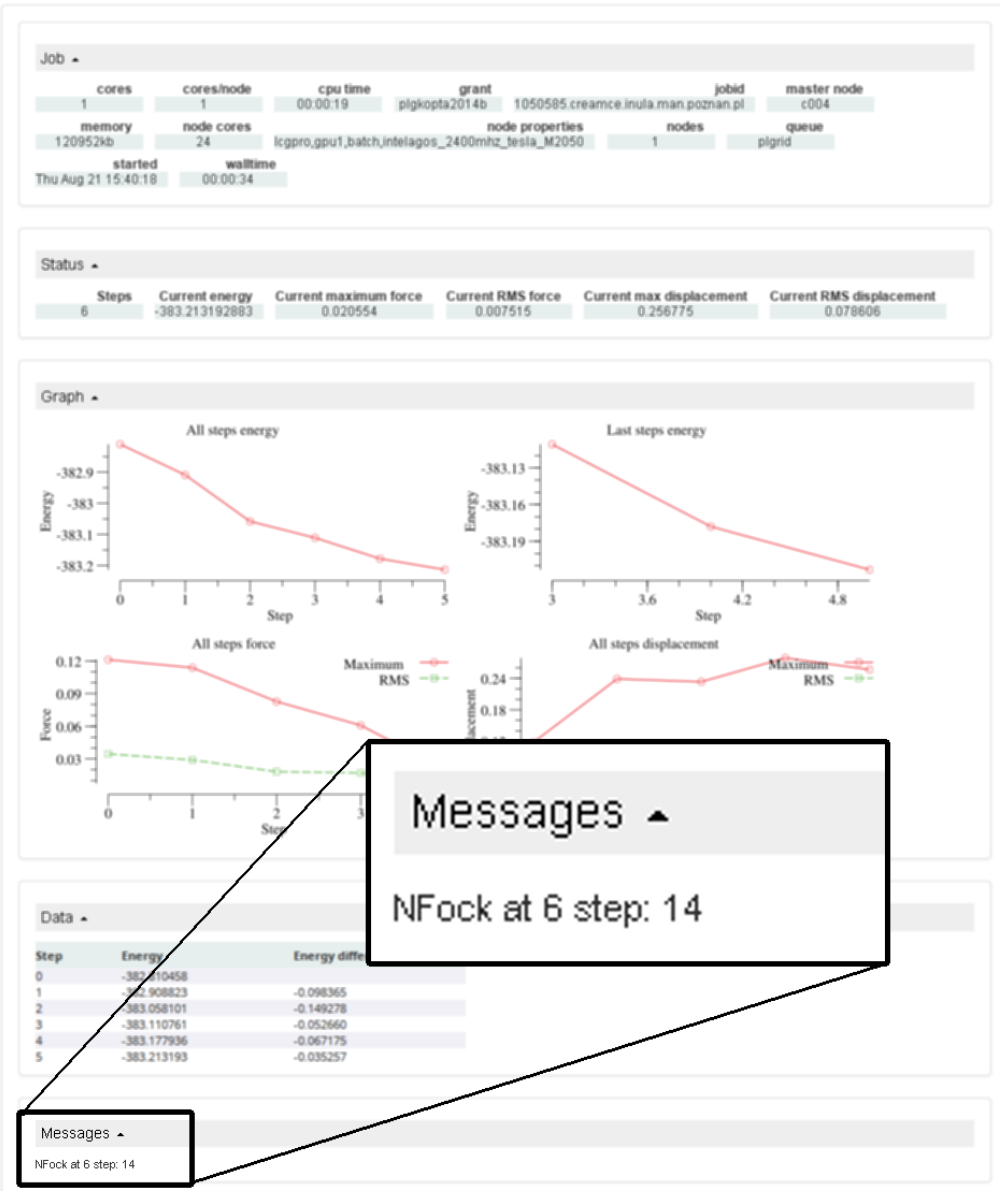
Poniżej umieszczony jest, wykorzystywany w zadaniu, skrypt powłoki *nfock.sh*.

```
#!/bin/bash

outfile=${QCG_OUTERR_FILE:-_stdouterr}
text=`awk '$1~/NFock/ { last=$2; nr+=1; } END { print "NFock at",nr,"step:",last }' $outfile`
if [ -n "$text" ]; then
        append_text "$text"
fi
```

Powyższy skrypt wykorzystuje program *awk* do wyszukania i policzenia wystąpień ciągu `NFock` w pliku ze standardowym wyjściem programu. Wartość ostatniego z tych wystąpień, wraz z opisem, jest przekazywana do funkcji pomocniczej `append_text`, której zadaniem jest zakodowanie jedej linii do formatu *JSON* w jakim dane są przesyłane ze skryptu monitorującego do serwisu QCG-Monitoring. Jeżeli tekst który chcemy przesłać wraz z danymi zawierałby więcej niż jedną linię, każda z nich musi być zakodowana funkcją `append_text`.

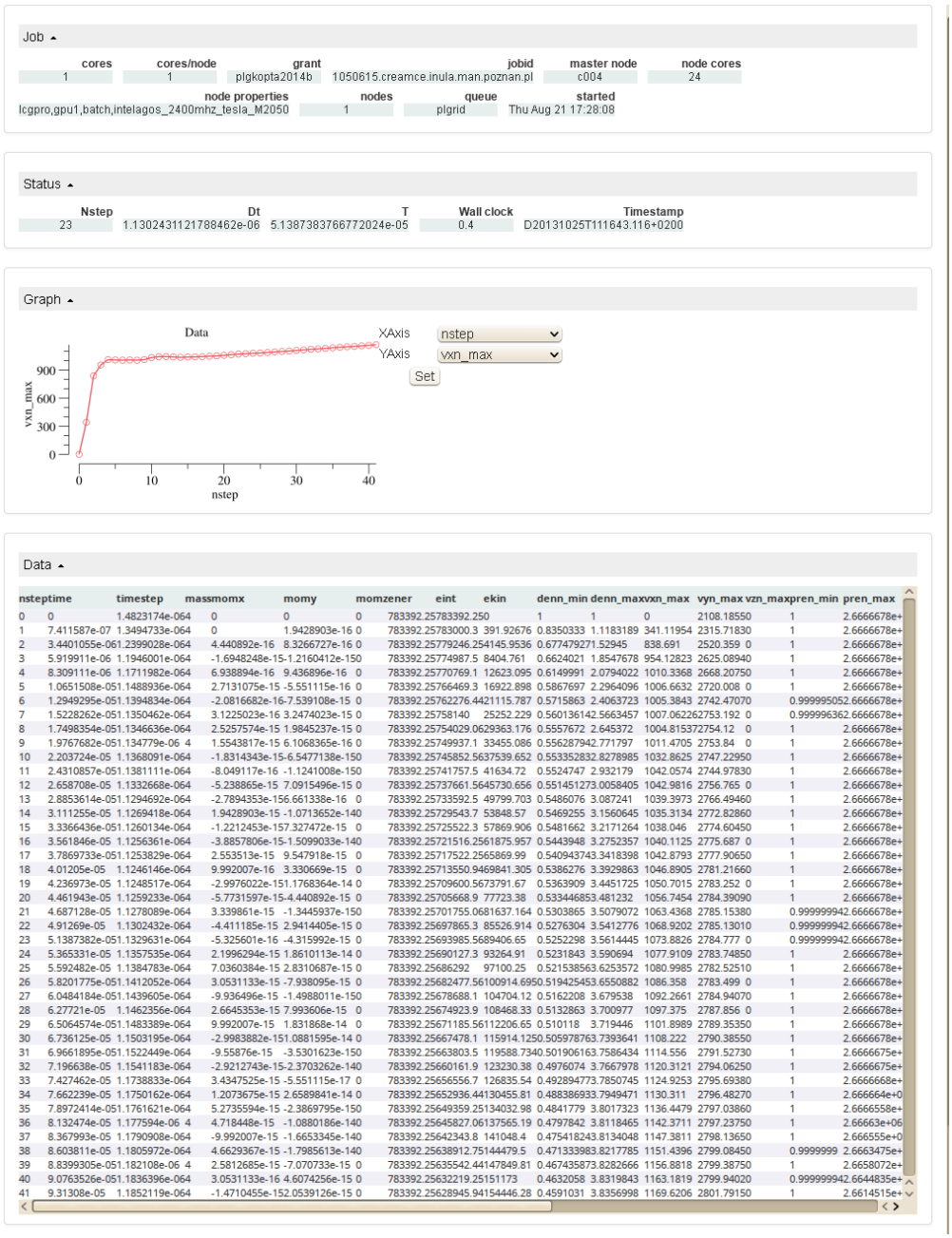
Aktualny stan zadania w serwisie QCG-Monitoring z wykorzystaniem powyższego opisu wygląda następująco.



piernik

Ten schemat monitorowania powstał dla aplikacji równoległej grupy użytkowników z dziedziny astro-fizyki, której czas działania liczony był w tygodniach. Prezentuje on następujące informacje:

- standardowa sekcja *Job* z parametrami środowiska uruchominiowego (zużycie procesora, nazwa grantu, lokalny identyfikator zadania, nazwa wezła, zadeklarowana pamięć, nazwa kolejki oraz czas trwania aplikacji),
- sekcja *Status* zawiera najważniejsze parametry symulacji,
- sekcja *Graph* zawiera wykres **aktualnie wybranych przez użytkownika** parametrów symulacji, możliwy jest wybór zarówno rzędnych, jak i odciętych wykresu,
- sekcja Data zawiera tabelaryczną postać parametrów symulacji ze wszystkich kroków czasowych.



Również ten schemat umożliwia użytkownikowi dodawanie własnych informacji tekstowych do prezentowanych danych. Mechanizm jest identyczny, jak dla schematu *gaussian*.

Ponieważ aplikacja użytkownika nie została zarejestrowana w systemie QCG, aby skorzystać z schematu monitorowania należy w opisie zadania wyspecyfikować nazwę schematu monitorowania:

```
#QCG monitor=piernik
```

Skrypty monitorujące

Skrypt monitorujący wykorzystywany przez schematy monitorowania, to skrypt powłoki który wypisuje na standardowe wyjście dane, przesyłane następnie za pośrednictwem mechanizmów *QCG*, do serwisu *QCG-Monitoring*. W serwisie tym, dane ze skryptu są transformowane oraz prezentowane użytkownikowi zgodnie ze schematem monitorowania. Dane przesyłane pomiędzy skryptem monitorującym a serwisem *QCG-Monitoring*, są zakodowane w formacie *JSON*.

Schemat danych *JSON* przesyłanych ze skryptu monitorującego zależy od schematu monitorowania. W poniższym punkcie przedstawimy schemat dla najczęściej przez użytkownika wykorzystywanego schematu *generic*.

Schema JSON dla schematu monitorowania generic

```
{
 "$schema": "http://json-schema.org/draft-03/schema",
 "description": "Schema for generic schema of QCG-Monitoring",
   "type": "object",
   "id": "http://jsonschema.net",
   "properties":{
       "job": {
           "type":"object",
           "properties":{
               "cores/node":      { "type":"string" },
               "cores":           { "type":"string" },
               "cpu time":        { "type":"string" },
               "grant":           { "type":"string" },
               "jobid":           { "type":"string" },
               "master node":     { "type":"string" },
               "memory":          { "type":"string" },
               "node cores":      { "type":"string" },
               "node properties": { "type":"string" },
               "nodes":           { "type":"string" },
               "queue":           { "type":"string" },
               "started":         { "type":"string" },
               "walltime":        { "type":"string" }
           }
       },
       "options": {
           "type":"object",
           "properties":{
               "overwrite":     { "type": { "enum": [ "true", "false" ] } },
               "showTimestamp": { "type": { "enum": [ "true", "false" ] } }
           }
       },
       "text": {
           "type":"array",
           "items": { "type":"string" }
       }
   }
}
```

Sekcja *job* jest wypełniana przez funkcję pomocniczą udostępnianą przez system QCG. Zawiera ona parametry środowiska wykonania pobierane z systemu kolejkowego.

Sekcja options może zawierać następujące właściwości:

- overwrite - jeśli wartość ustawiona jest na *true*, linie tekstu z tej wiadomości nadpiszą wszystkie dotychczas przesłane z zadania, wartość *fals* (domyślnie) spowoduje, iż nowy komunikat zostanie dopisany do listy starszych,
- showTimestamp - jeśli wartość ustawiona jest na *true*, linie tekstu z każdej wiadomości zostaną opatrzone etykietą w której będzie czas otrzymania tych wiadomości (domyślnie *false*).

Sekcja *text* zawiera tablicę ciągów znaków które mają być prezentowane w usłudze *QCG-Monitoring*.

Jeden element sekcji *text* musi być pojedyńczą linią. W przypadku gdy z zadania chcemy przesłać wiele linii, dla każdej z nich musi zostać utworzony element w tablicy *text*.

Poniżej prezentujemy przykładowy skrypt monitorujący dla schematu *generic*.

```
#!/bin/bash

function parse_text {
    echo "$*" | awk 'BEGIN { ORS=""; first=1 } { if (! first) { print ",\n" } print "\42"$0"\42"; first=0 }'
}

echo '{'

echo '"options": { "overwrite" : "false" }'
. ${QCG_APP_ROOT_DIR}/core/appmon/utils/monitor-common.sh
job_status > /dev/null 2>&1
if [ -n "$QCG_MONITOR_SECTION_JOB" ]; then
    echo ",\"job\": { $QCG_MONITOR_SECTION_JOB } "
fi

outfile=${QCG_OUTERR_FILE:-_stdouterr}
if [ -f "$outfile" ]; then
    out=$(grep " at " "$outfile" | tail -n 4)
    echo ', "text": [ '
    parse_text "$out"
    echo ']'
fi
echo '}'
```

Najważniejsze fragmenty powyższego skryptu:

- w linii 3 definiujemy funkcję która parametry wejściowe zakoduje jako tablicę typu *string* w formacie *JSON*,
- w liniach 10-11 włączamy definicję funkcji pomocniczej *job_status* służącej do pobrania informacji nt środowiska uruchomieniowego z systemu kolejkowego; funkcja ta zapisuje w zmiennej *QCG_MONITOR_SECTION_JOB* wynik swojego działania,
- w linii 16, na podstawie ustawień środowiska QCG, wybierana jest nazwa pliku zawierającego standardowe wyjście oraz standardowe wyjście błędów z uruchomionej aplikacji,
- w linii 18 wyszukujemy w pliku wyjściowym interesującego nas wzorca,
- w linii 20 używamy wcześniej zdefiniowanej funkcji *parse_text* do zakodowania znalezionych linii zawierających wzorzec zgodnie z formatem *JSON*.