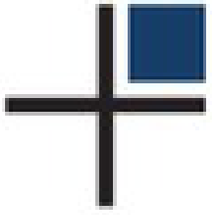
2ndQuadrant
Professional PostgreSQL

Implementare uno scheduler in User Space

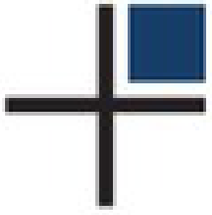
Gianni Ciolli

PGDay.IT 2014
Prato, 7 Novembre



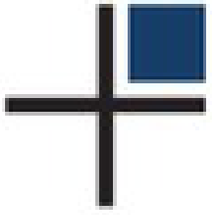
Sommario

- PGXN
- Estensione `worker_spark`
- Procedura in user space
- Esempio



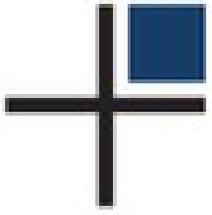
Custom Background Workers

- PostgreSQL 9.3+
- Implementare demoni
- Ciclo di vita “vicino” al server
- Migliore gestione!
- “mai più cron”



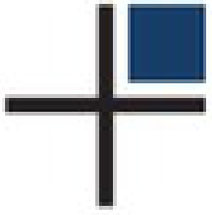
PGXN

- PostgreSQL Extension Network
- http://pgxn.org
- `pgxnclient`
  - `pgxn` in breve :-)



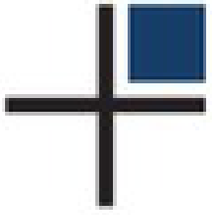
Estensione `worker_spark`

- Scritta in C
- Disponibile su PGXN



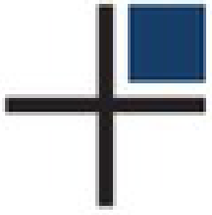
Ricerca su PGXN

```
$ pgxn info worker_spark
name: worker_spark
abstract: A background worker which regularly
 launches a procedure.
description: A background worker for
 PostgreSQL 9.3 which launches a given
 procedure in a given database at regular
 intervals.
maintainer: Gianni Ciolli <gianni.ciolli@2ndqu
license: PostgreSQL: http://www.postgresql.org
release_status: stable
version: 0.0.1
(...)
provides: worker_spark: 0.0.1
runtime: requires: PostgreSQL 9.3.0
```



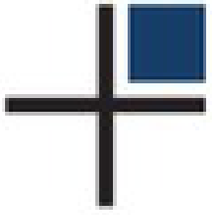
Scaricare il codice

```
$ pgxn download worker_spark
INFO: best version: worker_spark 0.0.1
INFO: saving
  /home/gianni/worker_spark-0.0.1.zip
```



Installazione e compilazione

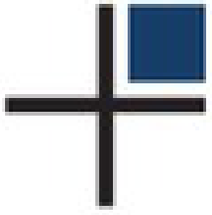
```
$ pgxn install worker_spark --sudo
INFO: best version: worker_spark 0.0.1
INFO: saving /tmp/tmpfUmYcs/worker_spark-0.0.1
INFO: unpacking: /tmp/tmpfUmYcs/worker_spark-0
INFO: building extension
gcc -g -O2 -fstack-protector --param=ssp-buffe
gcc -g -O2 -fstack-protector --param=ssp-buffe
INFO: installing extension
/bin/mkdir -p '/usr/lib/postgresql/9.3/lib'
/usr/bin/install -c -m 755
  worker_spark.so
  '/usr/lib/postgresql/9.3/lib/'
```



Parametri

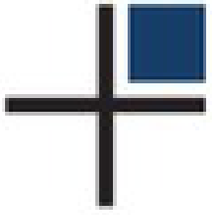
- Definiti dall’estensione
- Ogni estensione ha un proprio namespace
- Prefisso "`worker_spark.`"

```
worker_spark.database
worker_spark.naptime
worker_spark.procedure
worker_spark.schema
```



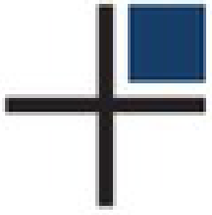
Parametri

```
postgres=# show worker_spark.database ;
 worker_spark.database
-----------------------
 postgres
(1 riga)

postgres=# show worker_spark.naptime ;
 worker_spark.naptime
----------------------
 5
(1 riga)
```



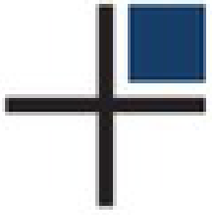
Parametri

```
postgres=# show worker_spark.schema ;
 worker_spark.schema
---------------------
 public
(1 riga)

postgres=# show worker_spark.procedure;
 worker_spark.procedure
------------------------
 user_space_scheduler
(1 riga)
```



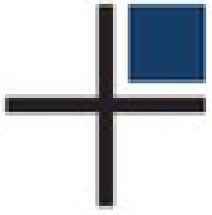
Procedura in user space

- Esempio “concorrente”
  - Anche se abbiamo solo 1 thread. . .
- PostgreSQL 9.4 introduce i background workers dinamici
  - Molto più complesso
  - Algoritmo “robusto”
  - Utile ginnastica :-)
  - “prevedere il futuro”. . .
  - Buona prassi o lusso?



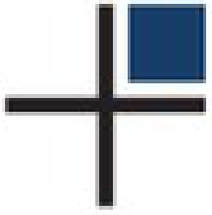
Procedura in user space 1/

```
BEGIN
  SELECT count(1)
  INTO v_procs
  FROM pg_stat_activity
  WHERE age(current_timestamp, query_start)
    > v_min_duration
  IF v_procs < v_max_procs THEN

    ...

  END IF;
END;
```



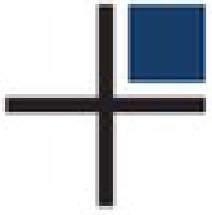
Procedura in user space 2/

```
FOR v_job_id, v_job_sql IN
   SELECT id, sql
   FROM jobs
   WHERE finish IS NULL
   ORDER BY priority
LOOP

   ...

END LOOP;
```



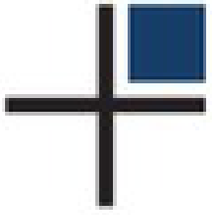
Procedura in user space 3/

```
BEGIN

  ...

EXCEPTION WHEN OTHERS THEN
  UPDATE jobs
  SET finish = clock_timestamp()
  ,   error  = SQLERRM
  WHERE id = v_job_id;
  EXIT;
END;
```

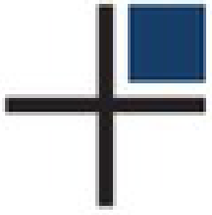


Procedura in user space 4/

```
IF pg_try_advisory_lock(v_job_id) THEN
  UPDATE jobs
  SET start = clock_timestamp()
  WHERE id = v_job_id;
  EXECUTE v_job_sql
  INTO v_outcome;
  UPDATE jobs
  SET finish  = clock_timestamp()
  ,   outcome = v_outcome
  WHERE id = v_job_id;
  EXIT;
END IF;
```

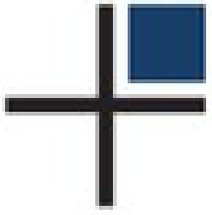


Esempio



E adesso. . .

Domande?



E infine. . .

Grazie per l’attenzione!

Licence