Gestione di flussi di dati GPS in near real-time

Alessio Siniscalchi <a.siniscalchi@bopen.eu>
http://it.linkedin.com/in/alessiosiniscalchi

B-Open Solutions – http://bopen.eu

B-Open solutions

Python dirige un'orchestra!

- architetture diverse
- dati eterogenei
- processamenti eterogenei
- debugging ed estensioni veloci
- interazioni con software 'legacy': utilizzo e controllo

Quasi tutto senza librerie esterne!

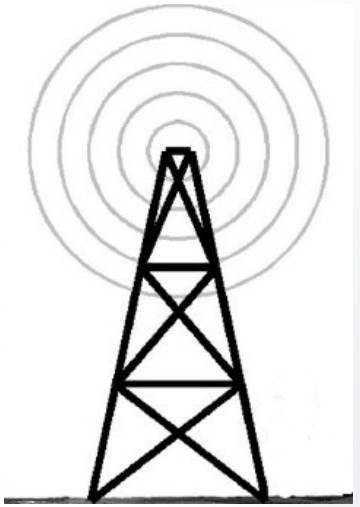
Sorgenti del flusso di dati

Monitoraggio di piattaforme (stazioni) in mare e on-shore

- Spostamenti plano-altimetrici N,E,Q

Canali di comunicazione inaffidabili

- Segnale GPS/GSM intrinsecamente instabile
- Risorse energetiche limitate a bordo



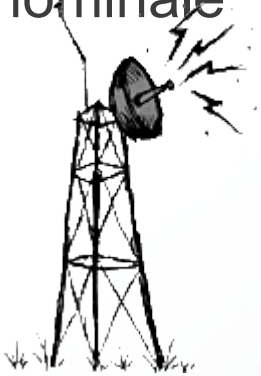
Emissione in near real-time

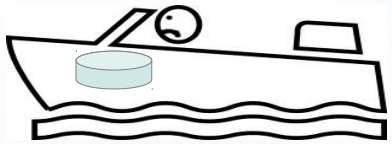
- Produzione continua, a blocchi orari
- Necessità di aggiornamento continuo dello stato
  - Limitato buffer di storage dei dati
  - Intrinseco rilievo dell'informazione dello spostamento

Dati di input

Formati eterogenei

- file RINEX (Receiver Independent Exchange Format) orari/giornalieri
- B-file, E-file, U-file giornalieri (binari)
- Sessioni orarie: ricerca stato 'nominale'





Processo di acquisizione dati

- manuale (operatore umano)
- semi-automatico (software comandati da operatore umano)

Storing dei dati di input

Ordini di grandezza:

- ~10MB/dì * 50 stazioni * 365 dì → ~ 180 GB/anno
- ~12 file/dì * 50 stazioni * 365 dì → ~ 213600 file/anno

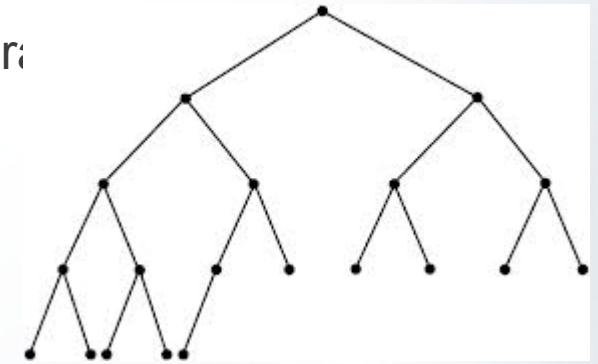
Storing eterogeneo a seconda dei software di preprocessamento:

- una cartella con file binari:

  nome file → stazione, tipo file, data, intervallo ora

- una struttura del tipo:

  stazione/anno/mese/giorno/ nome file



Storing non centralizzato:

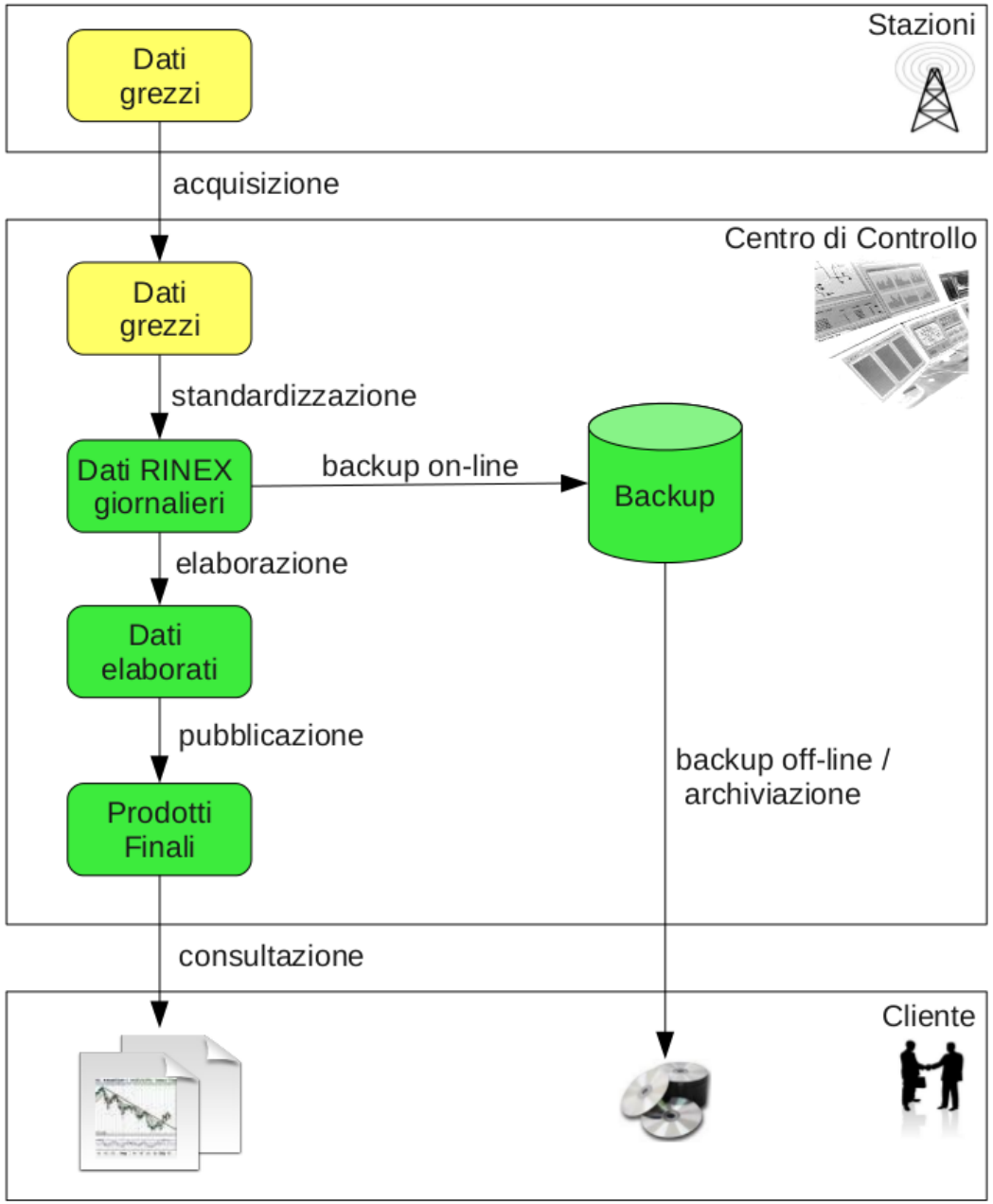
- diversi pc windows in rete attrezzati con hw/sw per il trattamento di un subset di stazioni

B-Open solutions

Lista dei desiderata

Acquisizione, Processamento, Pubblicazione, Reporting

- minimizzazione dell'intervento umano
- dati di output disponibili prima
- acquisizione e processamento ad alte prestazioni
- formato omogeneo RINEX giornaliero
- archiviazione centralizzata con backup
- reportistica di controllo rapido anomalie
- intranet di consultazione/interrogazione dati e grafici



B-Open solutions

Attivazione della catena

Attivazione singola, seriale:

+ il singolo processo conosce le risorse attive

- più processi in parallelo interferiscono fra loro

- I tempi sono lunghi

- una catena lunga può pregiudicare le prestazioni su tutte le stazioni

Attivazioni multiple, parallele:

- il processo ignora le risorse attive → gestione lock risorse attive

+ I processi non interferiscono fra loro

+ I tempi sono brevi

+ scalabilità

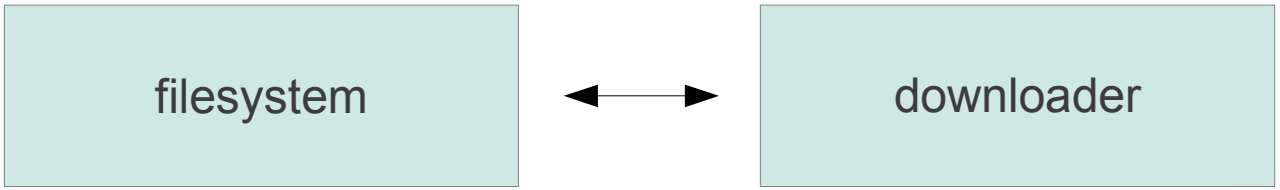
Gestione dei lock

https://pypi.python.org/pypi/lockfile

```
def lock_path(path, force=False, timeout=0):
    lock = LockFile(path)
    while not lock.i_am_locking():
        try:
            lock.acquire(timeout=timeout)
        except:
            if force:
                lock.break_lock()
                lock.acquire()
            else:
                raise
```

```
def unlock_path(path, force=False):
    lock = LockFile(path)
    if force:
        lock.break_lock()
    else:
        lock.release()
```

Step di acquisizione: entità



- tipologie stazioni
- dato più recente disponibile per stazione
- cartelle work in progress

No query alla stazione:
Quanti dati hai? Sei occupata ora?
Quando ti ho telefonato?

- tipologie stazioni
- hw libero/impegnato
- orario corrente
- ultima connessione effettuata per stazione
- misura efficacia download per stazione

- Da soppesare poche richieste di grossi pacchetti (-energia, +rischio connessione) a molte richieste di pacchetti piccoli (viceversa)

B-Open solutions

Step di acquisizione: apprendimento

Insuccesso: durante la richiesta di un gruppo di N file, questa parte ma non giunge al termine.

Decisione lunghezza ottimale dei “blocchi” di file: N

- Selezione statistiche rilevanti:
  - minimo numero di tentativi per un dato N (ad es. 2)
  - età della statistica più vecchia (ad es. 3 giorni)
- Senza statistiche: si è 'mediamente' ottimisti
- Variazione di N:
  - in caso di ultimo N con insuccesso: N = N//2 or 1
  - l'N più grande supera una certa % di successo (tarabile: es.70%): N += 1
  - Nessuna N supera quella soglia: N = 1
  - Altri casi: N rimane costante

Step di standardizzazione

Creazione Libreria di conversione verso RINEX giornaliero:

- utilizzo di directory di lavoro temporanee: tempdir
- lancio di eseguibili (teqc, upack12, XYZAshRx) per merging/conversione: popen, subprocess
- copie, rimozioni, spostamenti, ridenominazioni: os, shutil

Storing:

- Classe “StorageTree” per operare sul filesystem strutturato: datime, fnmatch, os.path
  - variabili di istanza per determinare lo stato dell'albero
  - metodi per operare in lettura/scrittura

Step di elaborazione

Estrazione informazioni di sintesi:

- generazione file intermedi (RINEX file “summary”) con teqc.exe (popen, subprocess)
- storing su csv di informazioni estratte dai file (indicatori sulle velocità di spostamento, entità dell'errore, etc.): generazione delle serie storiche di “Qualità” (datetime, csv)
- generazione delle serie storiche degli spostamenti (datetime, csv)
- pubblicazione di reportistica dello storing/backup: warning sull'assenza di file RINEX
  - DeltaCopy (http://www.aboutmyip.com/AboutMyXApp/DeltaCopy.jsp)
  - Chamaleon (https://pypi.python.org/pypi/Chameleon)

Aumento affidabilità

Controllo anti “impallaggio”: psutil (https://pypi.python.org/pypi/psutil)

```
def nostall_call(cmd, zero_cpu_timeout_sec,
logger=None, threshold=0, wait_before=60,
**popenkwargs):
```

- chiama *cmd* → attende *wait_before* → se la cpu rimane al di sotto di *threshold* per più di *zero_cpu_timeout_sec* con il processo acceso, uccide il processo

Controllo peso file generati: os.stat(path)[6]

Parsing dei log e dei std.out dei sw esterni

Step di pubblicazione e Consultazione

Sito intranet di pubblicazione risultati:

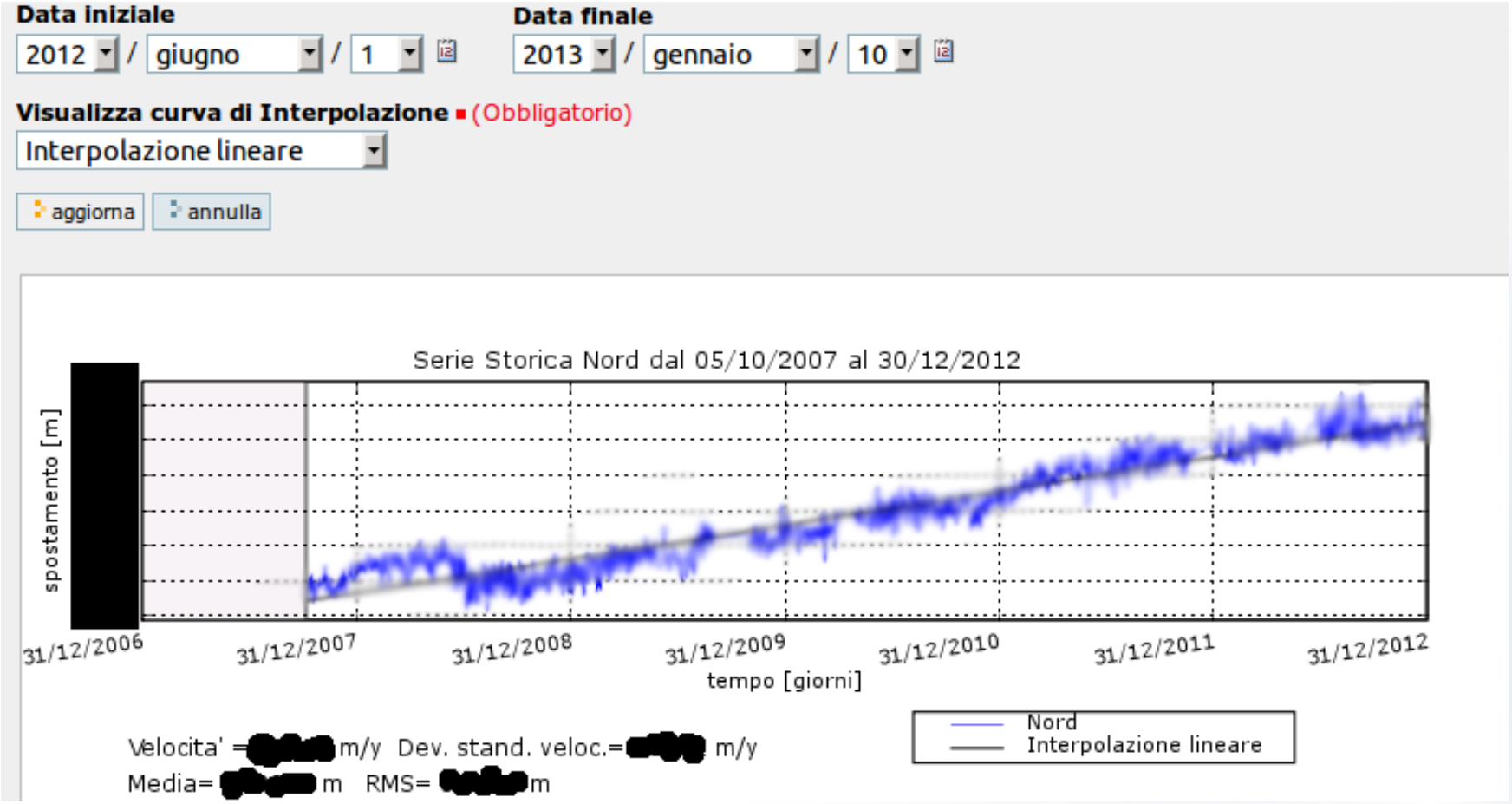
- Plone (www.plone.org) con prodotti (plugin) sviluppati ad-hoc
- uploader automatico dei file delle serie storiche sul sito
- generazione di grafici (matplotlib)

Consultazione

- upload/ricerca/download di documenti delle stazioni monitorate
- studio delle serie:
  - interpolazioni dei grafici
  - finestrature temporali
- interfaccia webgis per navigare fra le stazioni

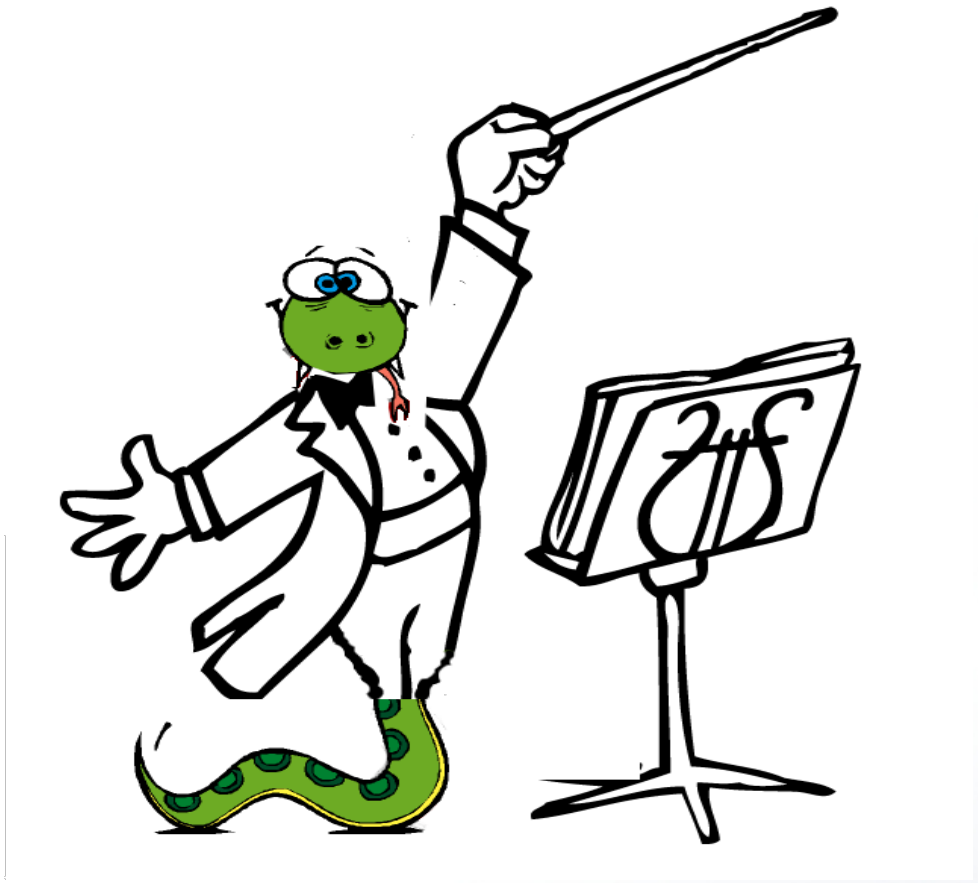
B-Open solutions

Visualizzazione e finestratura



B-Open solutions

Python ha diretto l'orchestra



Grazie.

Alessio Siniscalchi <a.siniscalchi@bopen.eu>

http://it.linkedin.com/in/alessiosiniscalchi

B-Open Solutions – http://bopen.eu

B-Open solutions