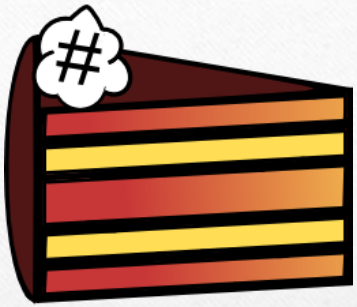


# Dessert

## Una libreria per la simulazione a eventi discreti in .NET


Relatore:

Dott. Giovanni Lagorio

Correlatore:

Prof. Giovanni Chiola

Candidato:

Dott. Alessio Parma

# Obiettivo principale

- Creare **clone** su .NET/Mono della libreria **SimPy**
  - Dedicata alla simulazione a eventi discreti (**DES**)
  - Scritta in, e consumabile da, codice **Python**
  - Estremamente semplice da usare, anche per non esperti
    - Tuttavia, le prestazioni **non** sono notevoli
- Nostro lavoro, **Dessert**, avrebbe dovuto:
  - Essere decisamente più **performante** di SimPy
  - Preservarne concetti principali e facilità d'uso

# Obiettivo secondario

- Realizzare **layer** di traduzione, **Armando**, con cui eseguire su Dessert simulazioni per SimPy
  - Senza cambiare una singola riga di codice
  - Potenziale (e gratuito) **incremento** delle prestazioni

# Scaletta della presentazione

1. **Pregi** e **difetti** di SimPy
   - Soprattutto, perché le prestazioni non siano ottimali
2. La nostra **soluzione** al problema
3. Il lavoro **preparatorio** per Dessert e Armando
4. Risultati ottenuti
5. Confronti e risultati a livello **prestazionale**

# Pregi di SimPy

- Leggibilità e **chiarezza**
  - Uso di Python
  - Ottime scelte di design
- **No multithreading**
  - Uso **coroutine** per **simulare** concorrenza
  - Processi a livello di simulatore
  - Codice più semplice e facile da scrivere
  - Forte **scalabilità**

```
>>> import simpy
>>>
>>> def clock(env, name, tick):
...     while True:
...         print(name, env.now)
...         yield env.timeout(tick)
...
>>> env = simpy.Environment()
>>> env.process(clock(env, 'fast', 0.5))
>>> env.process(clock(env, 'slow', 1))
>>> env.run(until=2)
fast 0
slow 0
fast 0.5
slow 1
fast 1.0
fast 1.5
```

# Difetti di SimPy

- Python è un'arma a doppio taglio
  - Codice **pulito** e leggibile, ma forti **penalità** nell'esecuzione
- Difficile raggiungere notevoli prestazioni
  - Linguaggio **dinamicamente tipato** e **interpretato**
  - Un **lock** globale nell'interprete **impedisce** di eseguire più simulazioni in **concorrenza**
  - **Rallentamenti** al crescere della scala delle simulazioni

# La nostra idea

- Cattive prestazioni dovute a linguaggio interpretato?
  - Riscrivere tutto su linguaggio **tipato** e **compilato**
- Necessità di preservare **design** basato su **coroutine**
  - Ricerca di un linguaggio che le possieda, **nativamente**
- Volontà di preservare design **concettuale** di SimPy
  - Nuovo linguaggio dovrà essere **object oriented**

# Ricerca dei linguaggi con coroutine

- Molti linguaggi «famosi» non le hanno **nativamente**
  - C, C++, Java le potrebbero avere solo tramite **estensioni**
- Altri (Go, Haskell, D, Erlang) le hanno
  - Ma quei linguaggi non ci erano **sufficientemente noti**
- Vari linguaggi per .NET (C#, F#, VB.NET) le hanno
  - **Esperienza** pregressa con molti di essi
  - Ottima scelta dal punto di vista **prestazionale**

# Valutazione degli elementi richiesti

- Motore del simulatore DES richiede **coda a priorità**
  - Per schedulare gli eventi nel tempo (**agenda**)
  - Struttura dati **non presente** nella libreria standard .NET
- Simulazioni richiedono generazione **numeri casuali**
  - Secondo diverse **distribuzioni** di probabilità
  - Libreria standard .NET **priva** delle classi richieste
- Simulatore deve essere **efficiente**
  - A livello **temporale** e **spaziale**

# Scrittura degli elementi necessari

# Libreria Dessert

## Dessert, esempio in F#

```
open Dessert // Yummy :P

let clock(env: IEnvironment, name,
          tick) = seq<IEvent> {
  while true do
    printfn "%s %f" name env.Now
    yield upcast env.Timeout(tick)
}

let env = Sim.NewEnvironment()
env.Process(clock(env, "fast", 0.5))
env.Process(clock(env, "slow", 1.0))
env.Run(until = 2)
```

## SimPy, esempio in Python

```
import simpy

def clock(env, name, tick):
  while True:
    print(name, env.now)
    yield env.timeout(tick)

env = simpy.Environment()
env.process(clock(env, 'fast', 0.5))
env.process(clock(env, 'slow', 1.0))
env.run(until=2)
```

# Caratteristiche di Dessert

- Libreria usabile da ogni linguaggio per .NET
  - Lavoro fatto per garantire **CLS compliancy**
- Libreria **multipiattaforma**
  - Usabile sia .NET, sia su Mono
- **Compatibilità concettuale** con SimPy
- Possibilità di eseguire più simulazioni in **parallelo**

# Armando, layer di traduzione

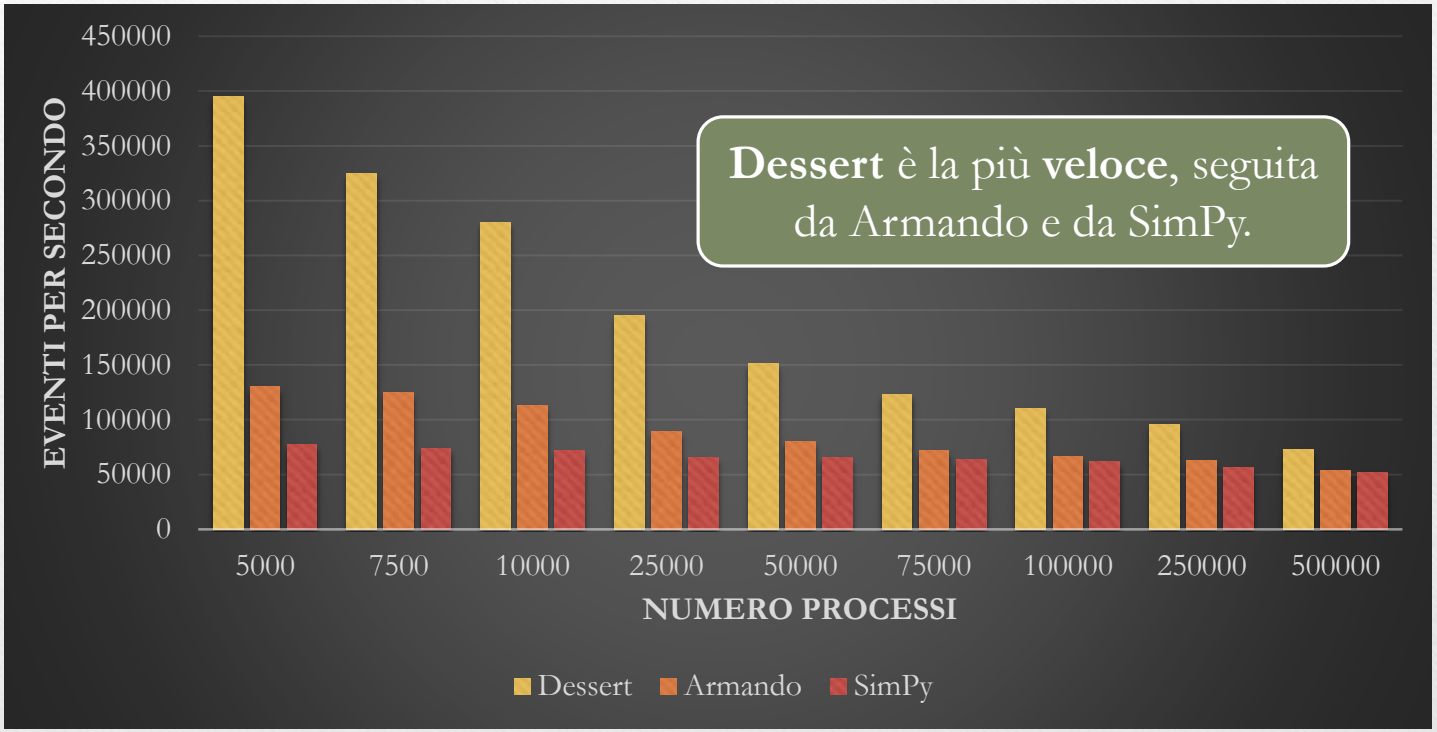
- Esegue simulazioni per SimPy su Dessert
  - Garantisce **compatibilità sintattica**
  - **Nessuna modifica** ai sorgenti delle simulazioni
  - **Potenziale aumento** delle prestazioni
- Abbondante uso della libreria IronPython
  - Per eseguire gli script Python
  - Per connettere il codice Python con quello .NET
- Richiede ulteriore lavoro di **raffinamento**
  - Per migliorare la **gestione degli errori** e la **compatibilità**

# Confronto tecnico
# Descrizione del benchmark

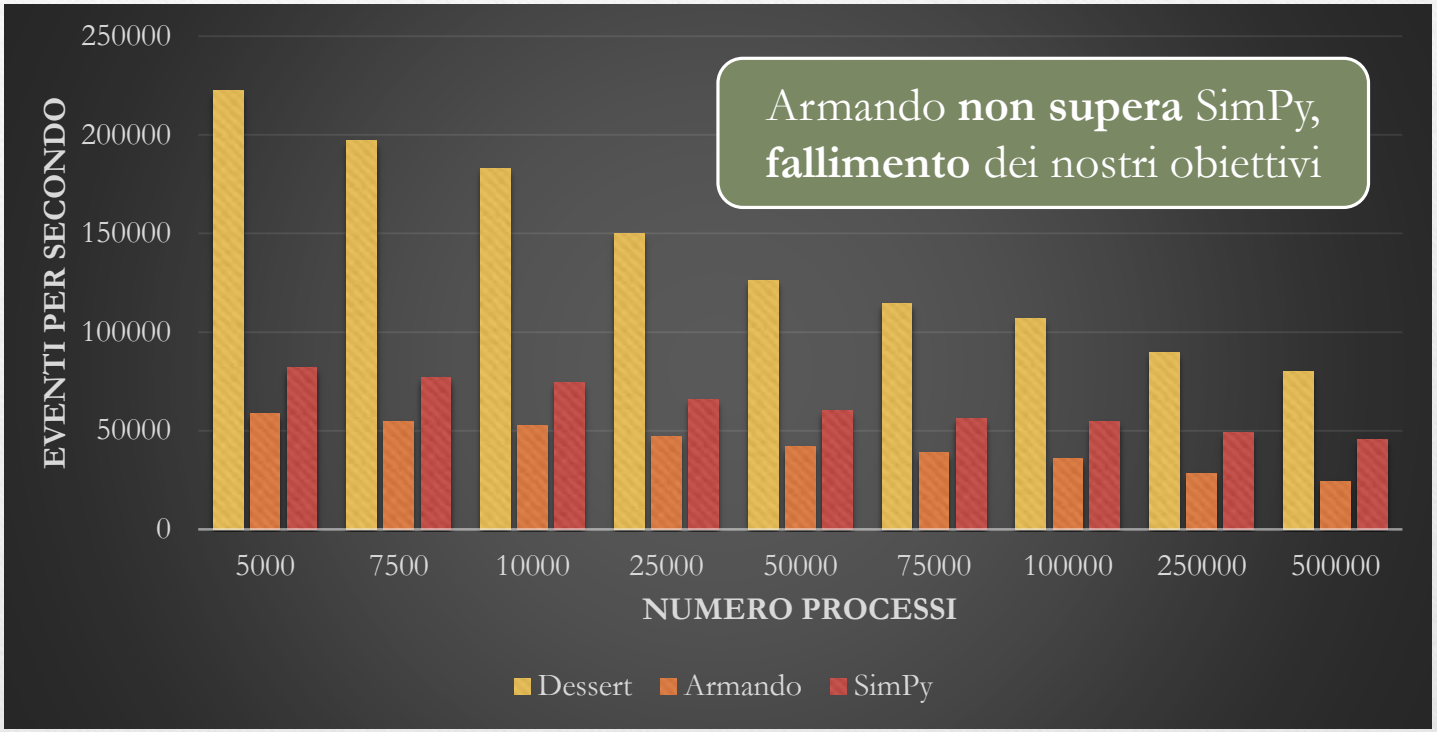
- **Stress** del simulatore
  - Creazione di molti processi, da 5000 a 500000, definiti dal codice a lato
- Misura velocità **processing**
  - Eventi elaborati per secondo
- Sia su Windows, sia su Linux

```
def gilberto(env, counter):
    while True:
        delay = counter.randomDelay()
        yield env.timeout(delay)
        counter.increment()
```

# Confronto tecnico
# Risultati su Windows

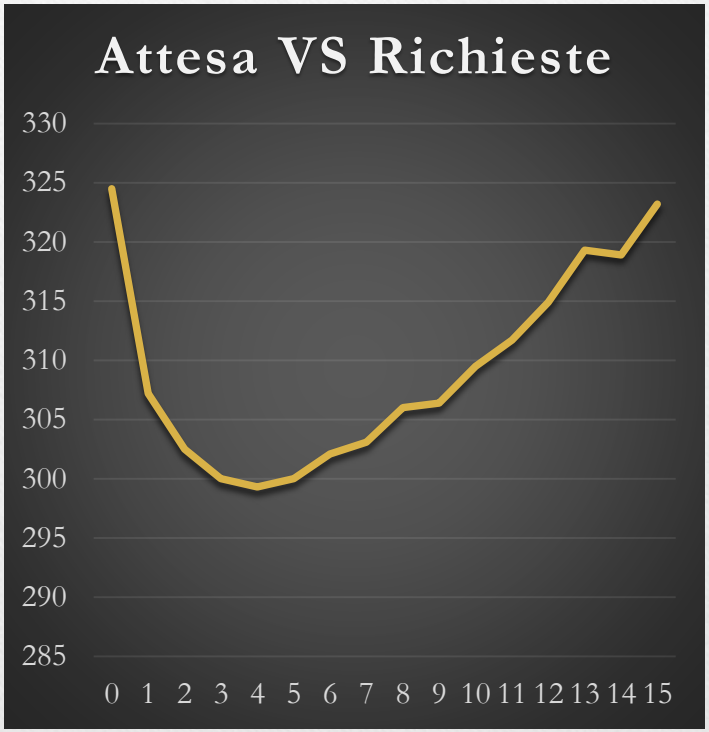


# Confronto tecnico
# Risultati su GNU/Linux

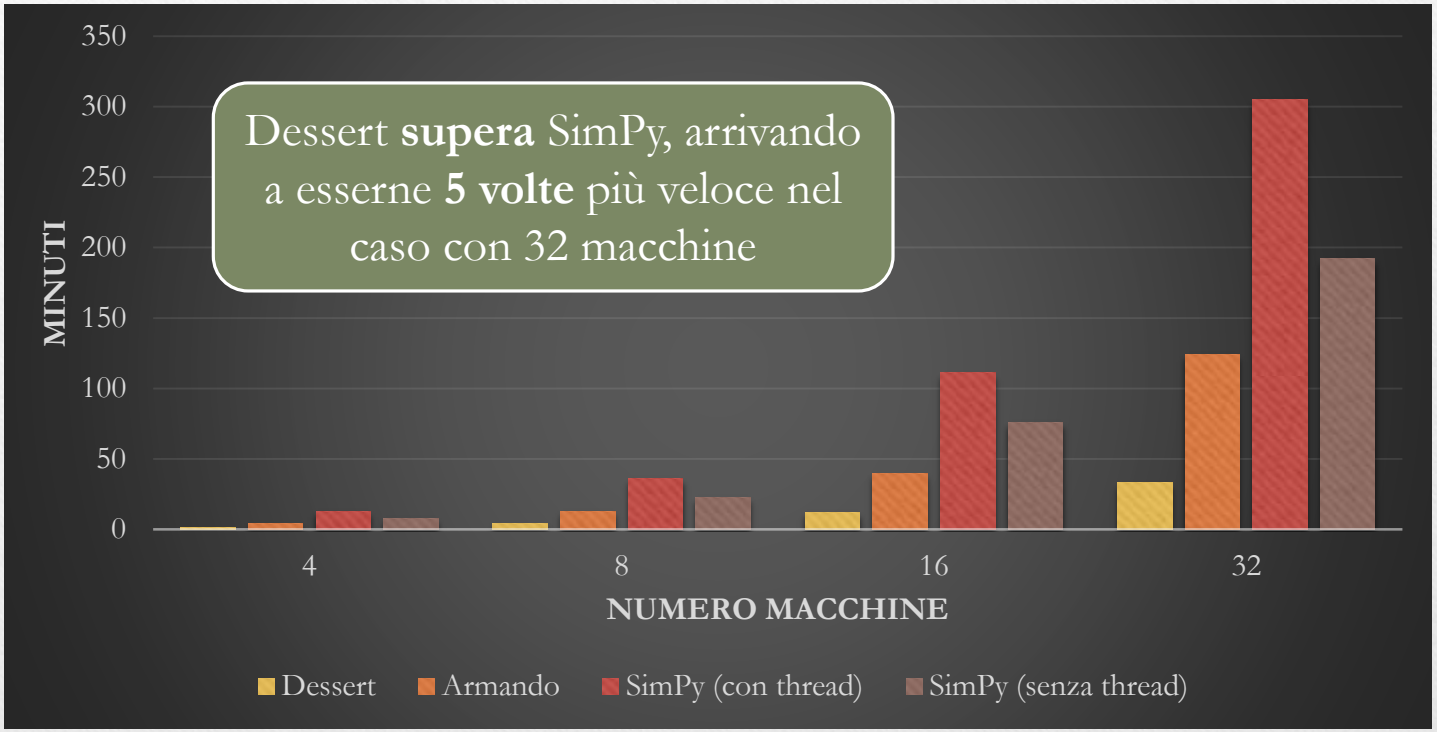


# Confronto realistico
# Descrizione del benchmark

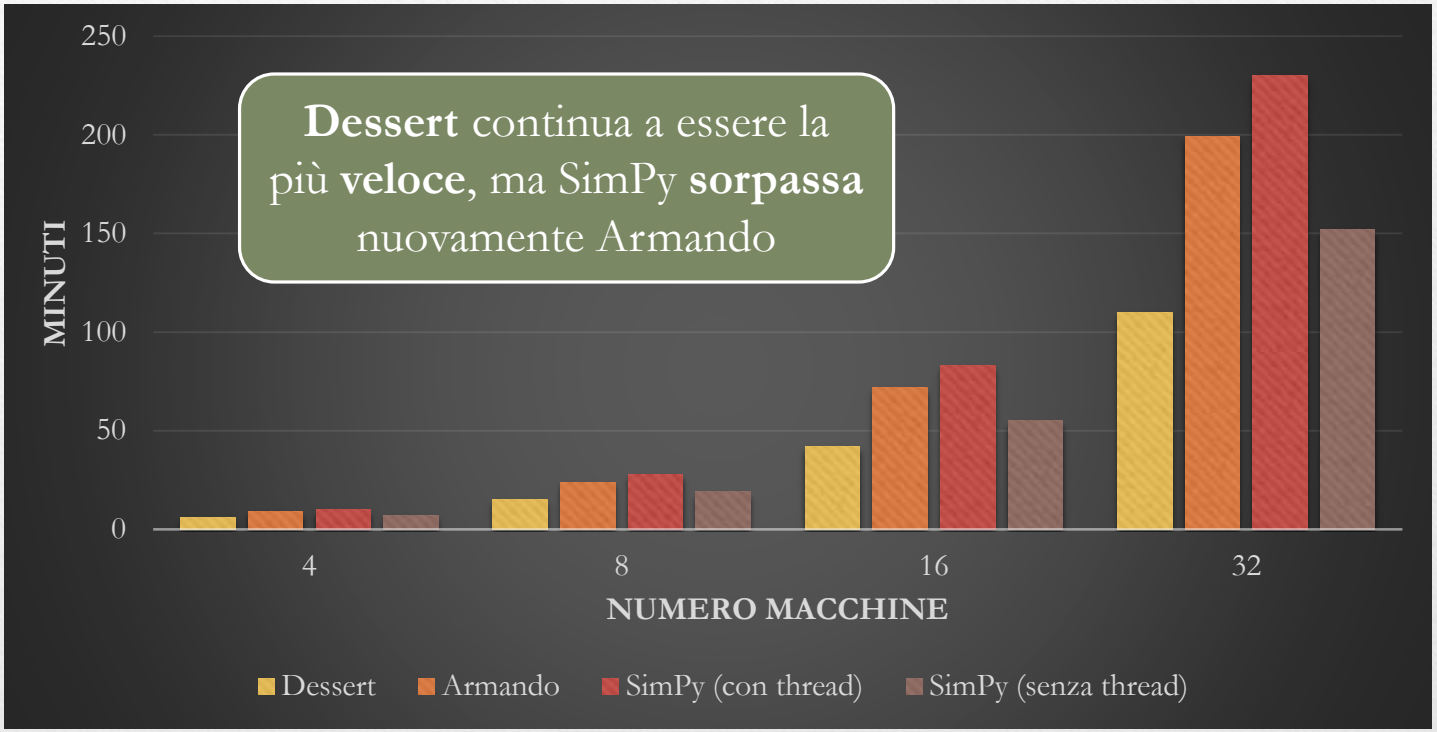
- Simula una rete **peer to peer**
  - **Interazioni** tra diversi client, server e uno switch
  - Ricerca numero richieste ideale per non **saturare** rete
- **Analisi** del sistema variando le macchine attive
  - Da 4 a 32 macchine
  - Simulazioni su più **thread**
  - Più test per ciascuna **configurazione**



# Confronto realistico
# Risultati su Windows



# Confronto realistico
# Risultati su GNU/Linux



# Riassumendo…

- Realizzata una libreria, Dessert, per la DES su .NET
  - Più **veloce** di SimPy, pur essendone **conforme**
  - Usabile da un maggior numero di linguaggi
- Sperimentato con IronPython, creando un **layer** con cui eseguire le simulazioni SimPy su Dessert
  - Più veloce di SimPy, ma per ora soltanto su Windows
- Scritti esempi, test e benchmark per verificare **correttezza** ed **efficienza** del nostro lavoro

Presentazione conclusa ☺

# Grazie per l'attenzione!

E grazie anche per non aver riso **troppo** di fronte

agli **intelligentissimi** nomi che sono stati scelti

per varie parti del progetto, primo fra tutti **Armando**!

# Domande?

Poche e non cattive, mi raccomando ☺