Implementazione di algoritmi in linguaggio “Python” per lo sviluppo di modelli 3-D QSAR tramite l’interfacciamento tra UCSF Chimera e il portale www.3d-qsar.com

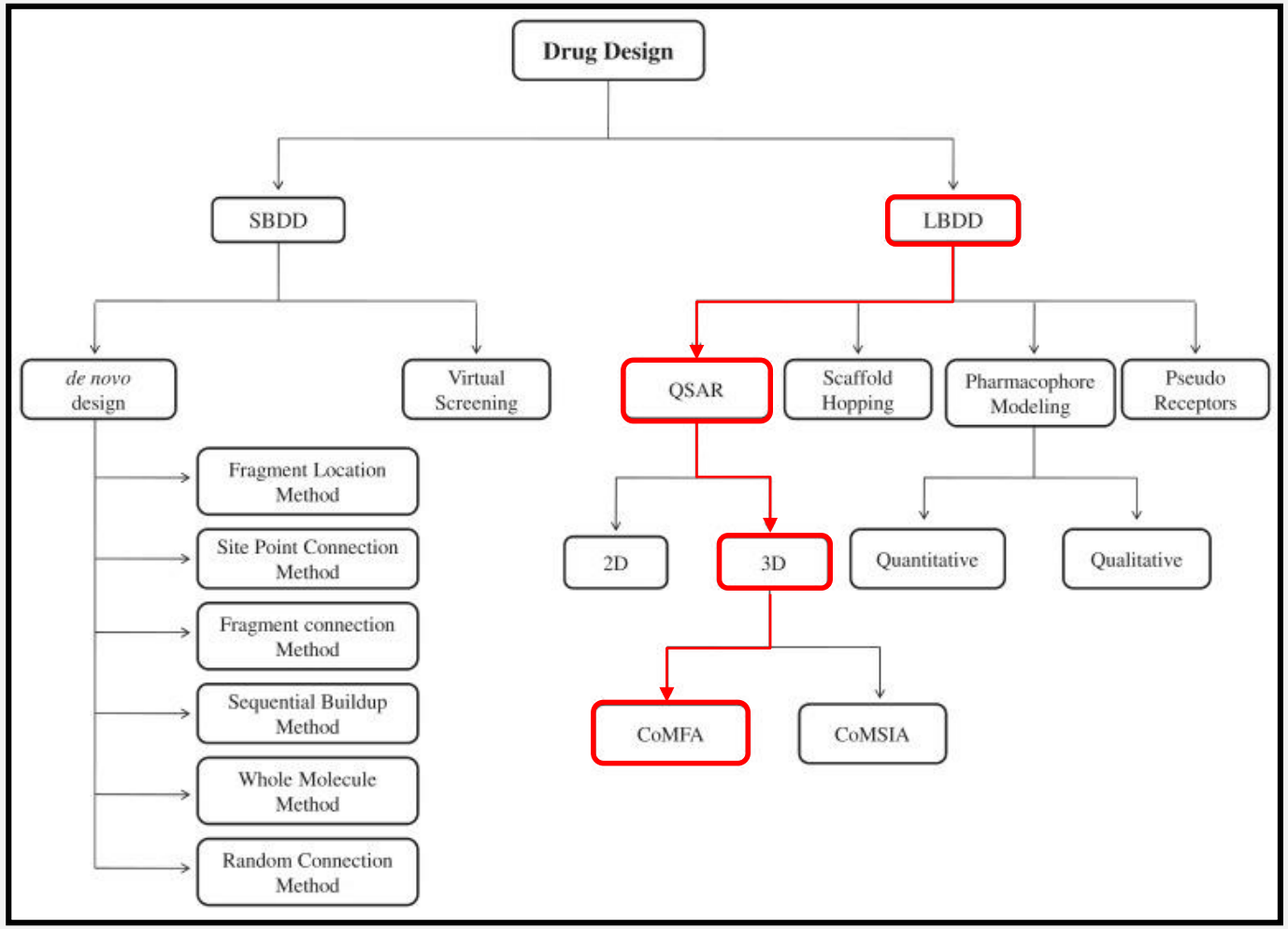
SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Farmacia e Medicina
Corso di laurea magistrale in Biotecnologie Farmaceutiche
Tesi di laurea sperimentale in Chimica Farmaceutica

Relatore:
Prof. Rino Ragno

Candidato:
Nikola Dino Capocchiano
Mat: 1199265

Introduzione

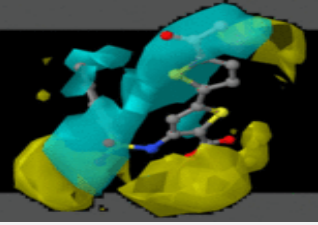


Ligand-**B**ased **D**rug **D**esign

Quantitative **S**tructure-**A**ctivity **R**elationships

3-**D**imensional

Comparative **M**olecular **F**ield **A**nalysis



3-D QSAR

Disegno della struttura 2D della molecola

Marvin Sketch

Trasformazione in struttura 3D

Marvin Sketch / Balloon

Generazione dei conformeri

Balloon

Allineamento dei vari conformeri

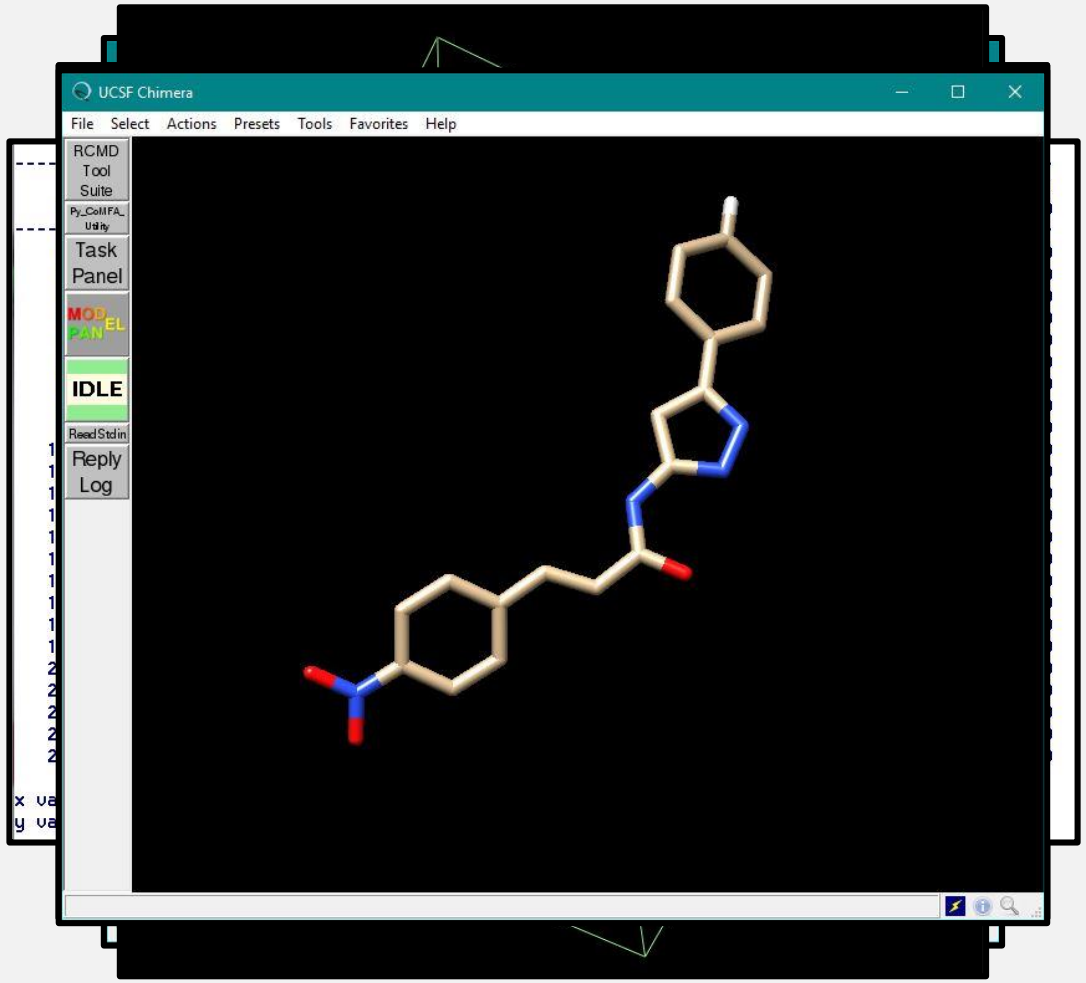
ShaEP

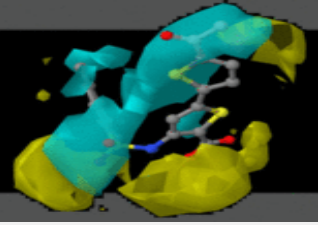
Generazione del modello 3-D QSAR

Py-COMFA

Analisi delle Mappe CoMFA

Chimera

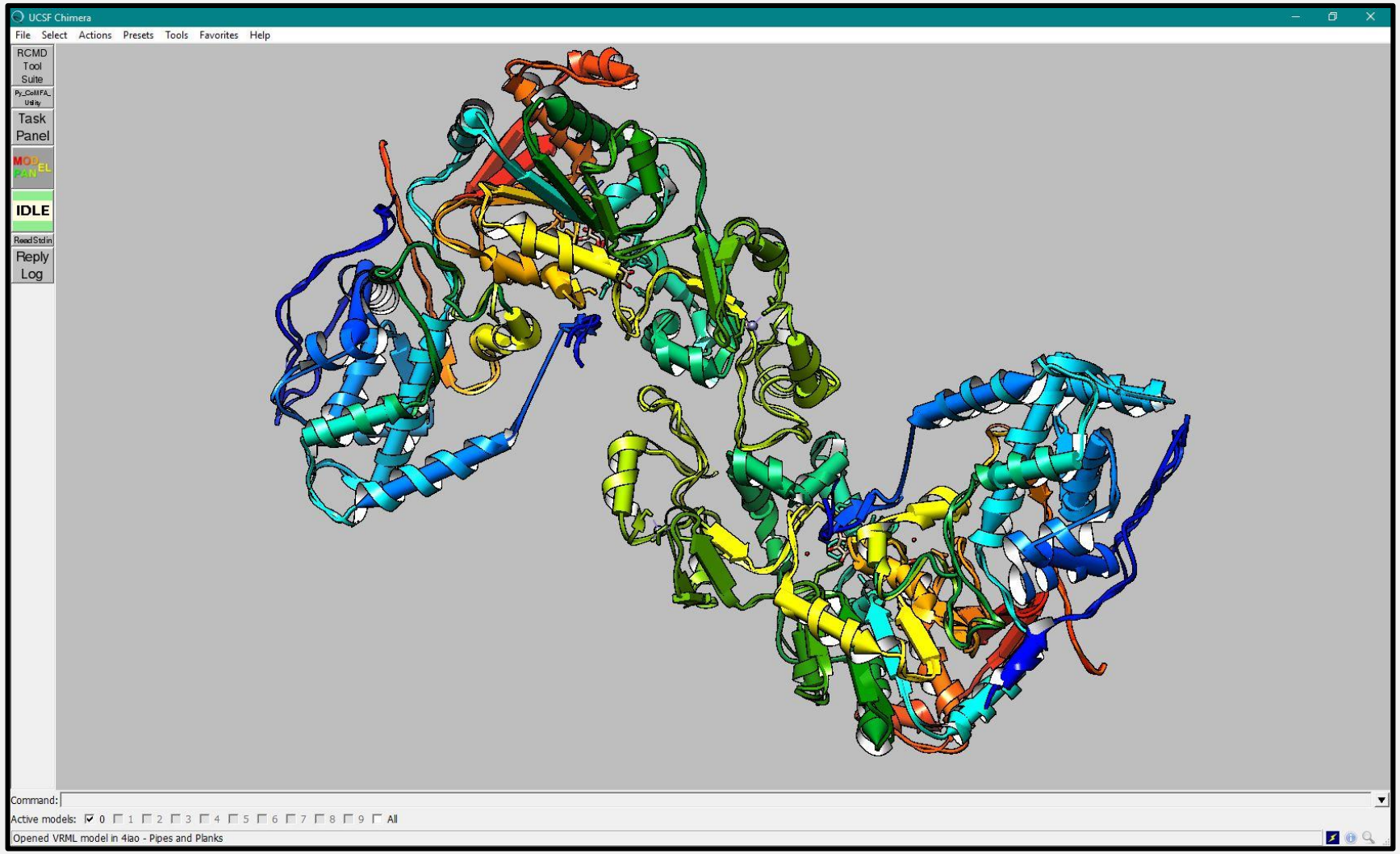


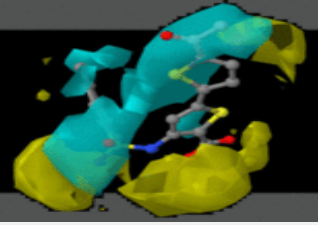


Punti Critici

1. Interfaccia Utente Unificata
2. Prestazioni
3. Esecuzione Remota
4. Facilità Di Utilizzo

UCSF Chimera

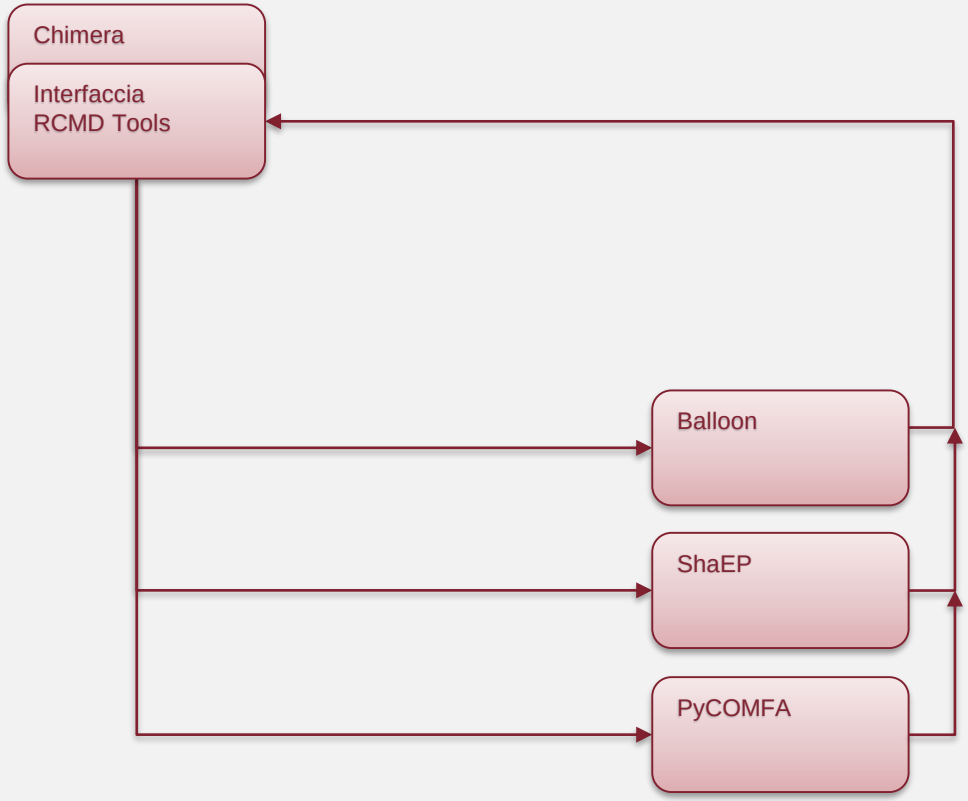




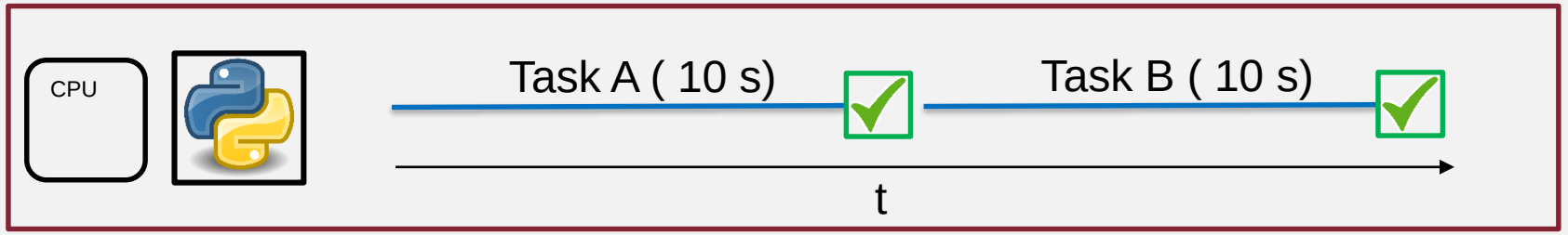
Python

- **Python is:**
- **A high-level programming language**
- **Interpreted**
- **Interactive**
- **Object-Oriented**

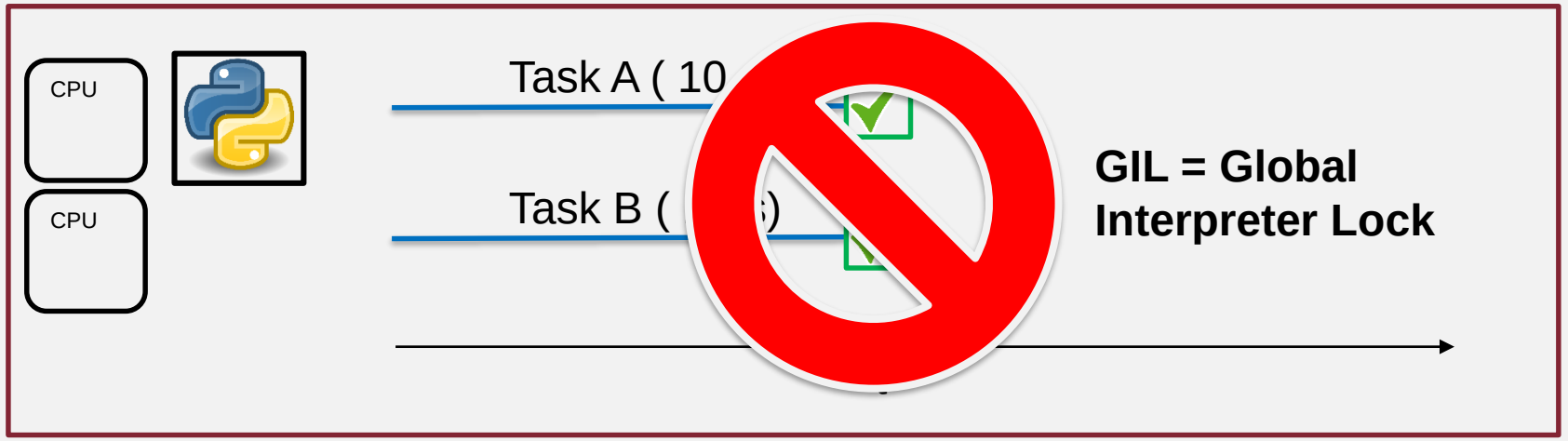
Organizzazione Plug-in



Prestazioni

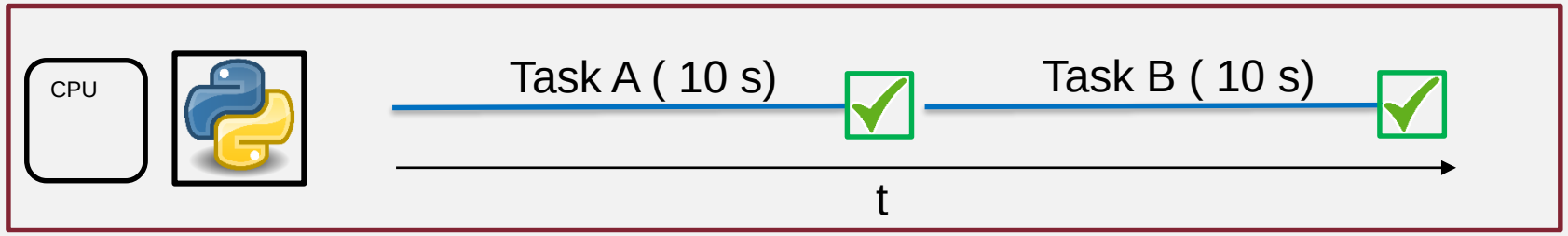


t_{Finale} = A + B = 20 secondi

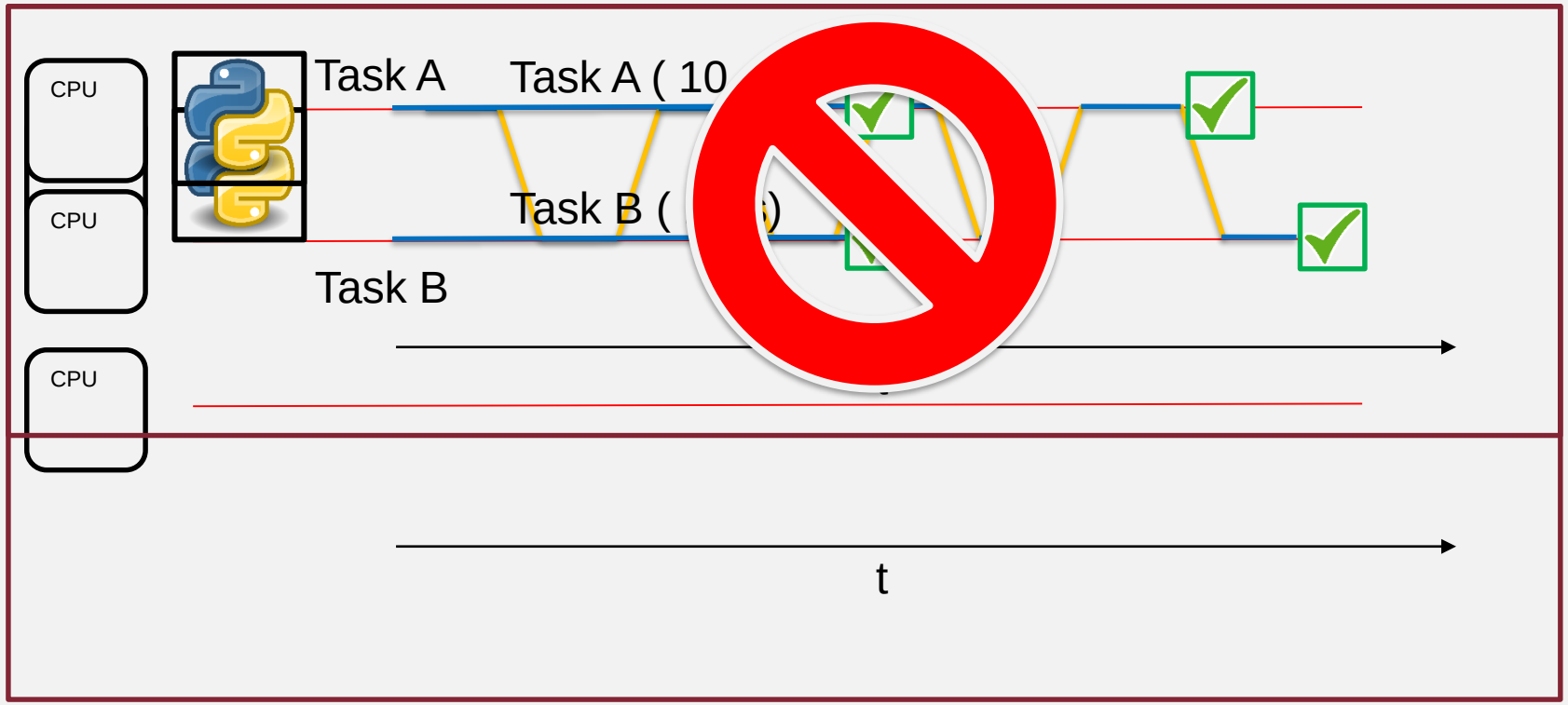


t_{Finale} = A = B = 10 secondi

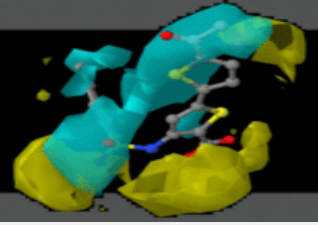
Prestazioni



t_{Finale} = A + B = 20 secondi



t_{Finale} = ?



Prestazioni

```
def count(n):
    while n > 0:
        n -= 1


numero = 90000000


a = datetime.datetime.now().replace(microsecond=0)


count(numero)
count(numero)


b = datetime.datetime.now().replace(microsecond=0)


tempo_finale= b - a
print(«Done in: » + str(tempo_finale))
```

Funzione: Conta da valore X a 0

X = 90 000 000

Tempo Finale = 7 secondi

(3.5 secondi a ciclo)

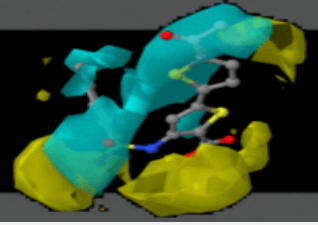
hiamalo A

equenzialmente

Prendi il tempo in secondi, chiamalo B

Tempo finale = B - A

Stampa tempo finale



Prestazioni

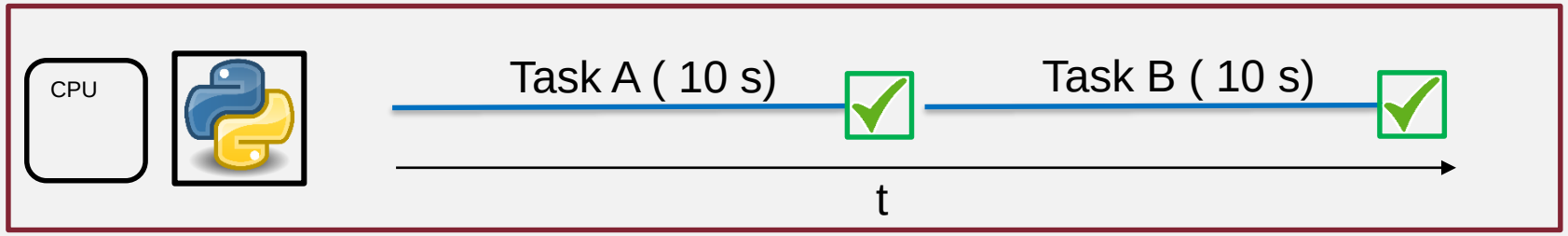
```
def count(n):
    while n > 0:
        n -= 1

numero = 90000000

a = datetime.datetime.now().replace(microsecond=0)

t1 = threading.Thread(target=count, args=(numero,))
t1.start()
t2 = threading.Thread(target=count, args=(numero,))
t2.start()
t1.join(); t2.join()

b = datetime.datetime.now().replace(microsecond=0)

tempo_finale= b - a
print("Finito in:" + str(tempo_finale))
```

Tempo Finale = 14 secondi

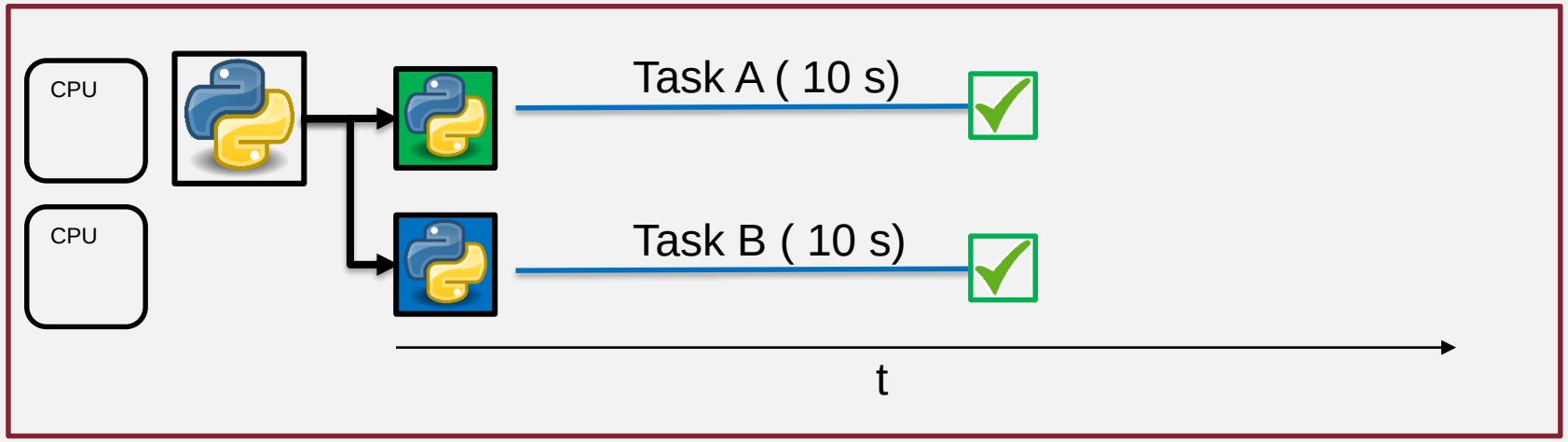
Overhead di 7 secondi!

eseguono 1

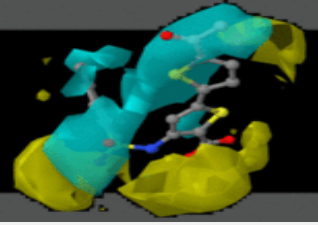
Prestazioni



$t_{Finale} = A + B = 20$ secondi



$t_{Finale} = A = B = 10$ secondi



Prestazioni

```
def count(n):
    while n > 0:
        n -= 1
    print "done"


if __name__ == '__main__':
    numero = 90000000
    a = datetime.datetime.now().replace(microsecond=0)

    p = multiprocessing.Process(target=count, args=(numero,))
    n = multiprocessing.Process(target=count, args=(numero,))

    p.start();n.start()

    p.join(); n.join()

    b = datetime.datetime.now().replace(microsecond=0)
    tempo_finale = b - a

    print("Finito in:" + str(tempo_finale))
```

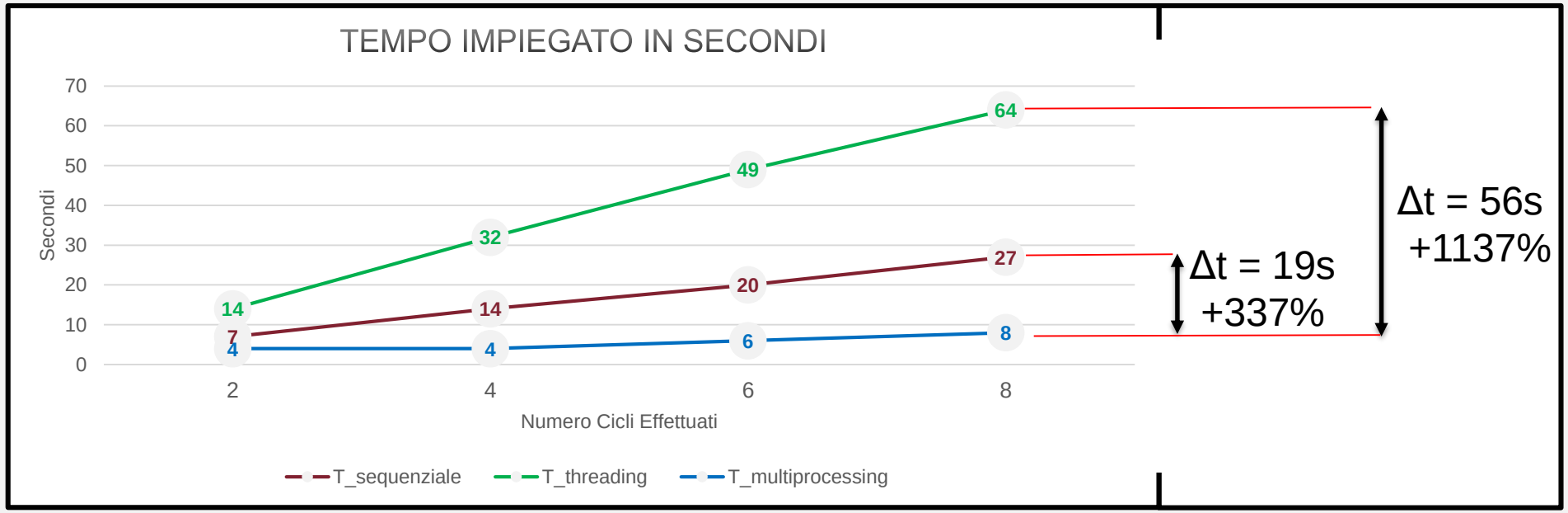
Tempo Finale = 4 secondi

Crea 2 **processi** differenti che eseguono 1 ciclo della funzione ciascuno

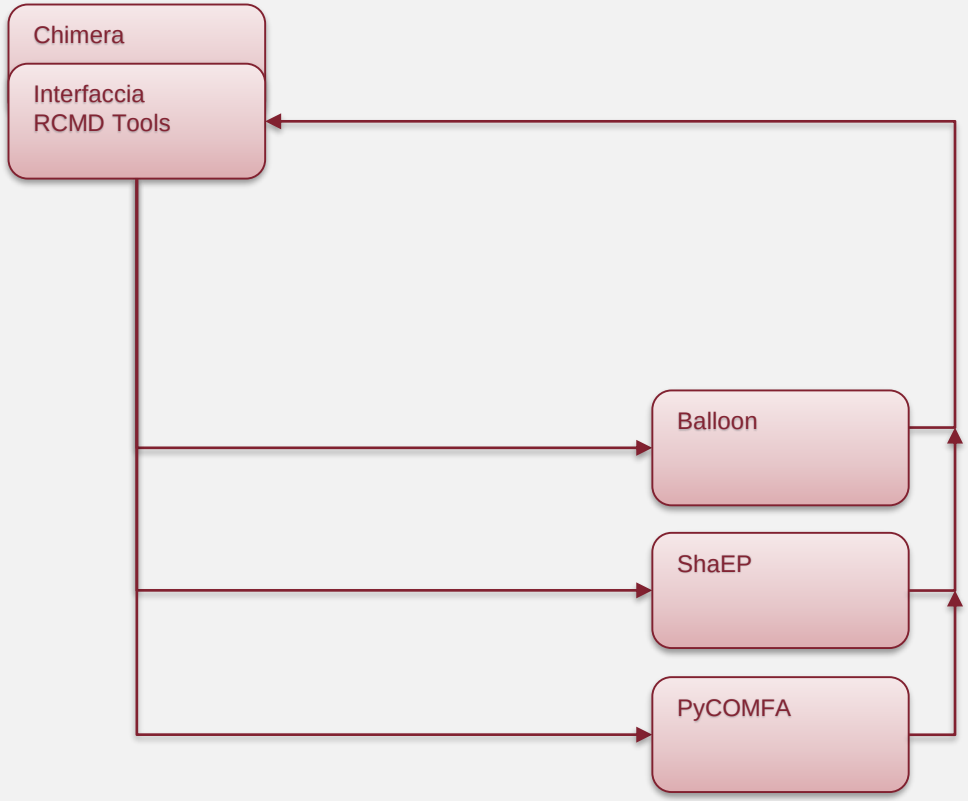
Prestazioni – Dati Conclusivi

CPU a 4 Core

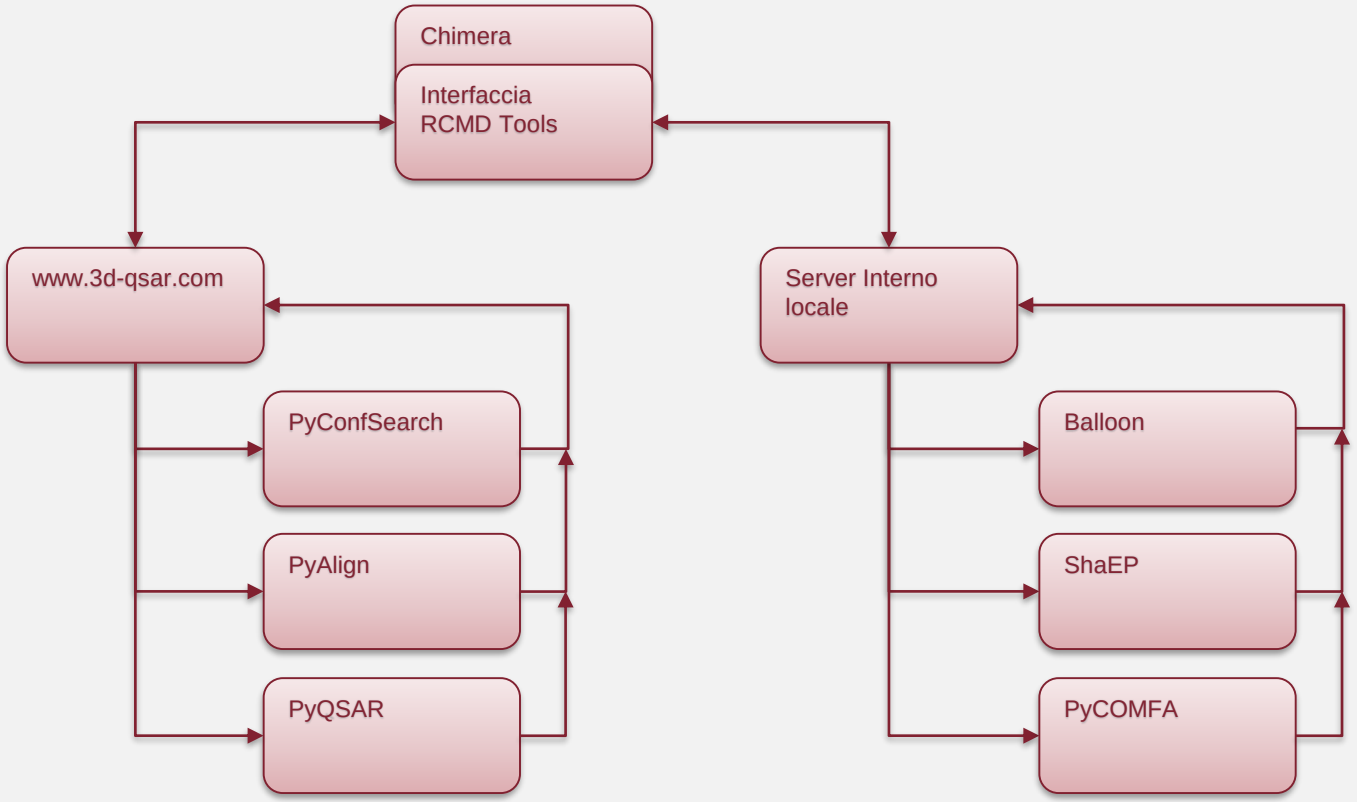
Numero di Cicli	T_sequenziale	T_threading	T_multiprocessing
2	7 secondi	14 secondi	4 secondi
4	14 secondi	32 secondi	4 secondi
6	20 secondi	49 secondi	6 secondi
8	27 secondi	64 secondi	8 secondi



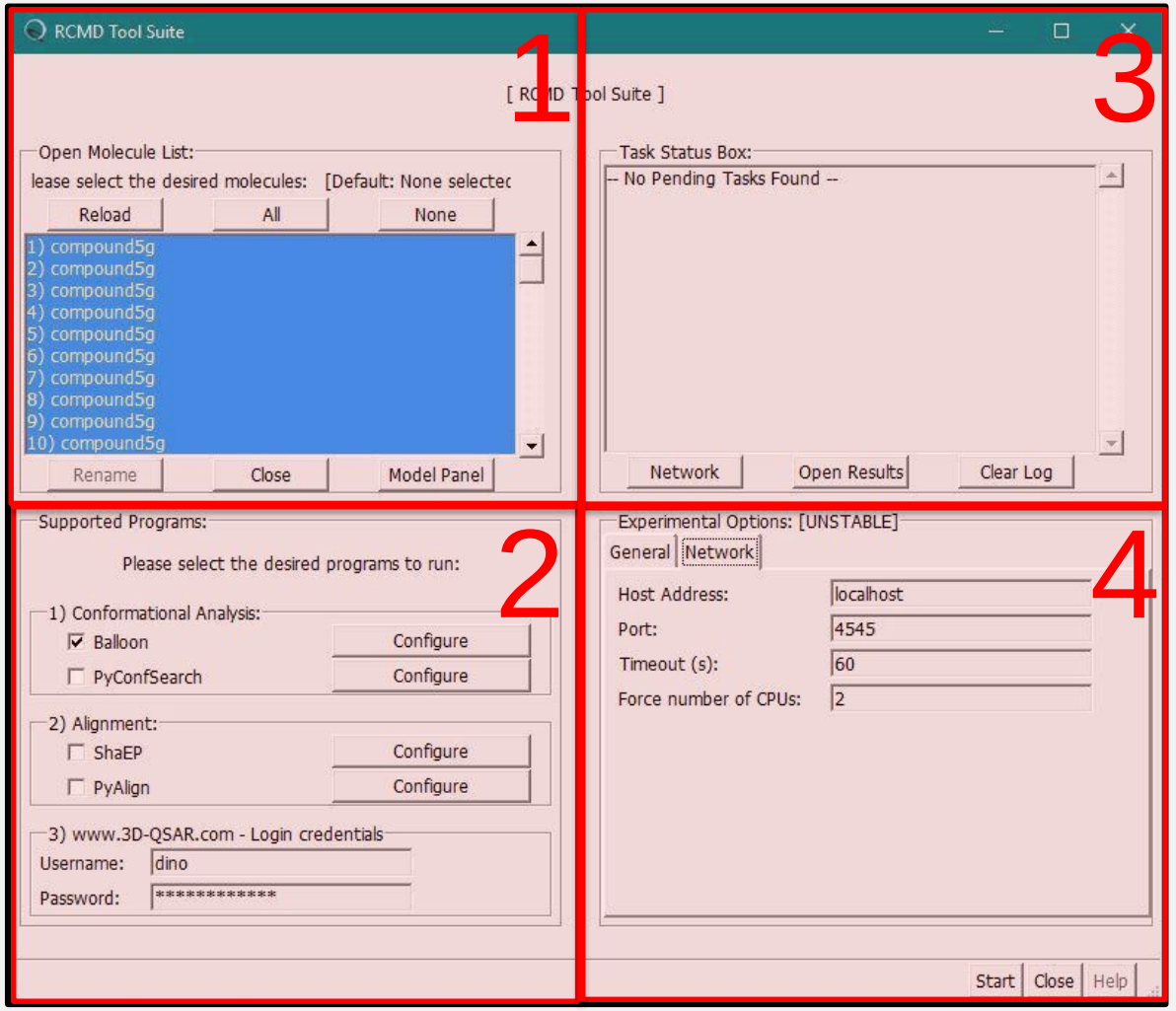
Organizzazione Plug-in



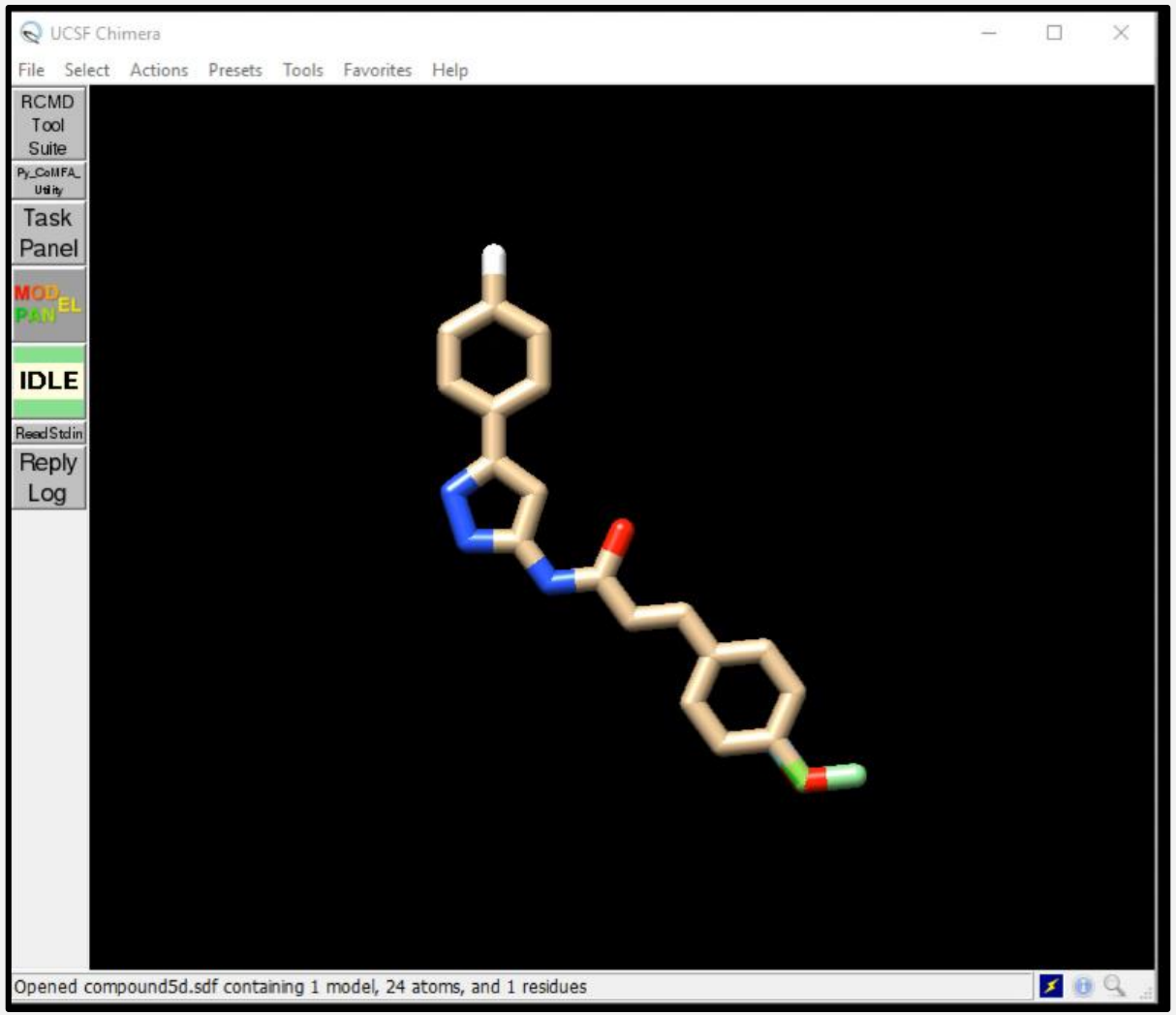
Organizzazione Plug-in

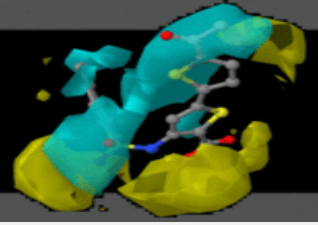


Interfaccia Principale



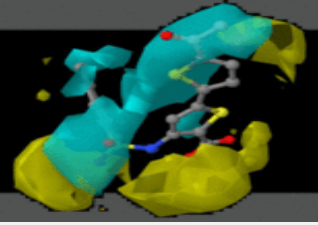
Esempio di utilizzo





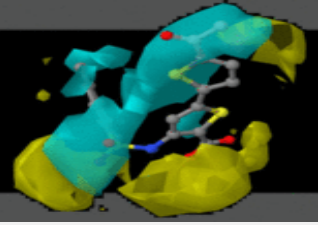
Punti Critici

1. Interfaccia Utente Unificata
2. Prestazioni
3. Esecuzione remota
4. Facilità di utilizzo



Punti Critici

1. Interfaccia Utente Unificata
2. Prestazioni
3. Esecuzione remota
4. Facilità di utilizzo



Conclusioni

- Risoluzione dei 4 punti critici presi in considerazione
- Possibilità di espansione a Chromebook e Cellulari
- Altamente modulare e di facile mantenimento
- Nuovo strumento didattico