PARALLEL SERVER WITH BERNSTEIN PHASE TYPES

NOTE PER UN ELABORATO QESM – 21 MAGGIO 24

EV, LS

Sommario

1 CONTESTO TECNOLOGICO

Abbiamo un pool di Containers che servono richieste che arrivano con processo di Poisson e tempo di servizio Bernstein Phase Type.

E' possibile combinare classi multiple dei tempo di servizio, che potrebbe essere il caso di un container che ospita endpoint diversi per uno stesso bounded context.

Il parametro da dimensionare è il PoolSize, che conduce ad un tradeoff fra il costo del numero di Containers attivi e il numero di invocazioni che vengono rifiutate.

La distribuzione del tempo di servizio può essere basata su osservazioni e fitting attraverso campionamento nei pesi di un BPH

TBD: endpoints diversi nel POD potrebebro avere necessità di scalare diversamente …. Che conduce verso una situazione in cui devi separare microservizi coesivi pe ril bene della scalabilità

2 SVILUPPO

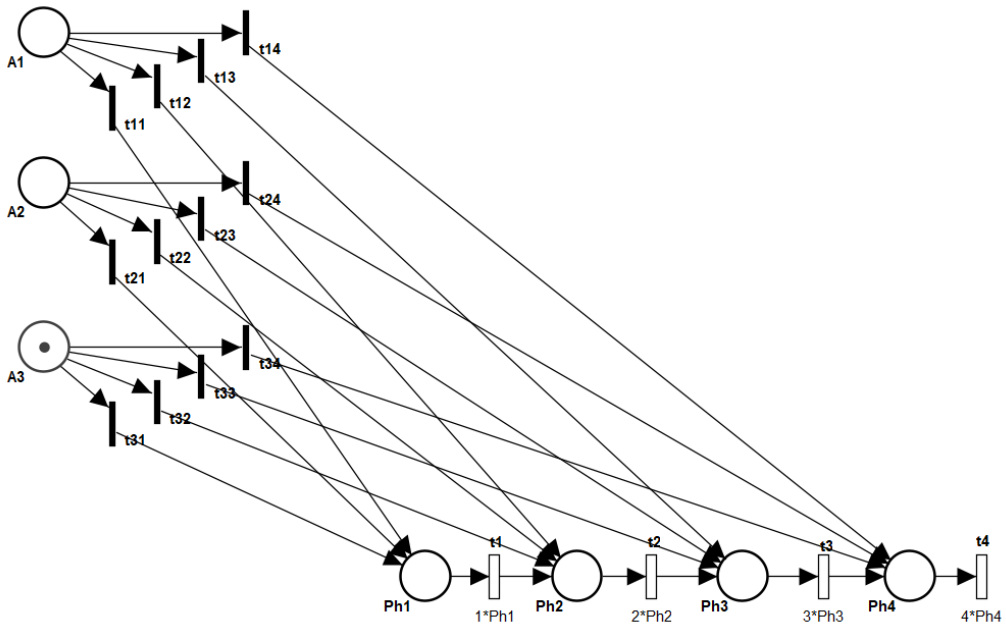
È possibile generare automaticamente il generatore infinitesimale, e su quello risolvere direttamente analisi steady state e transiente … su cui poi diventa possibile ottimizzare la dimensione del Pool secondo un modello di costo del disservizio (vincolato da SLA) e del costo di avere il Container attivo ma idle.

E' possibile valutare la differenza di operazione rispetto al caso in cui il tempo di servizio è stimato con distribuzione Exp anziché BPH

3 MODELLO

3.1 DISTRIBUZIONE CDF DEL BPH

(parallelServerBernsteinPh_BphCdfMultiplClasses.xpn)

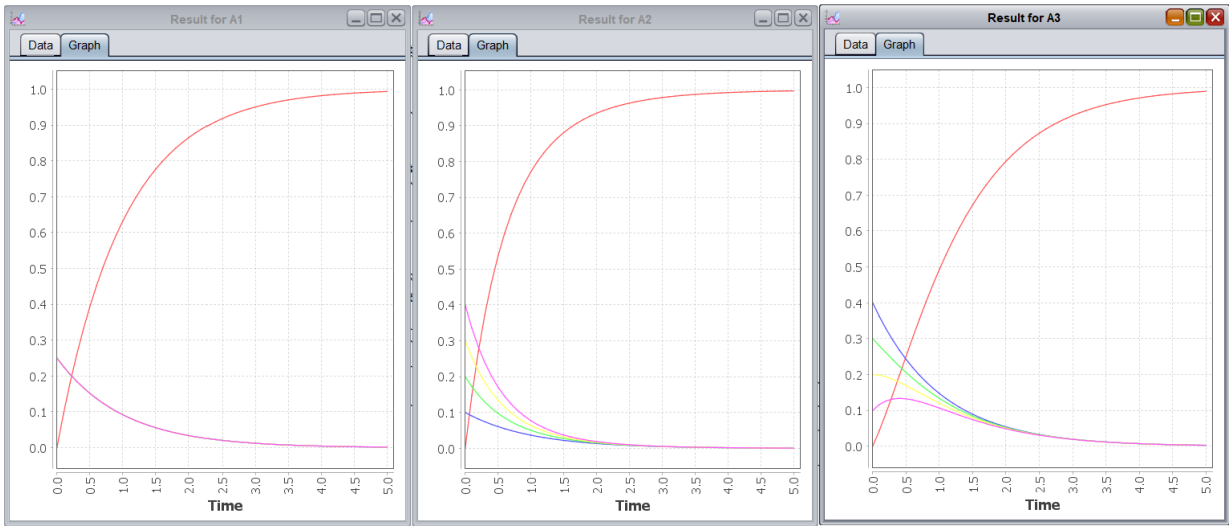


I 3 blocchi attivati da un token in A1, A2, A3, si distinguono per gli weights associati al random switch delle transizioni immediate in uscita, che determinano la fase di ingresso nel BPH Ph1,Ph2,Ph3,Ph4.

Nel blocco in alto (A1) i pesi sono 1,1,1,1, in quello intermedio (A2) sono 1,2,3,4, in quello in basso (A3) sono 4,3,2,1. I tre casi hanno diverso valor medio, dove 1,2,3,4 ha valore medio piu' basso, e 4,3,2,1 il più alto.

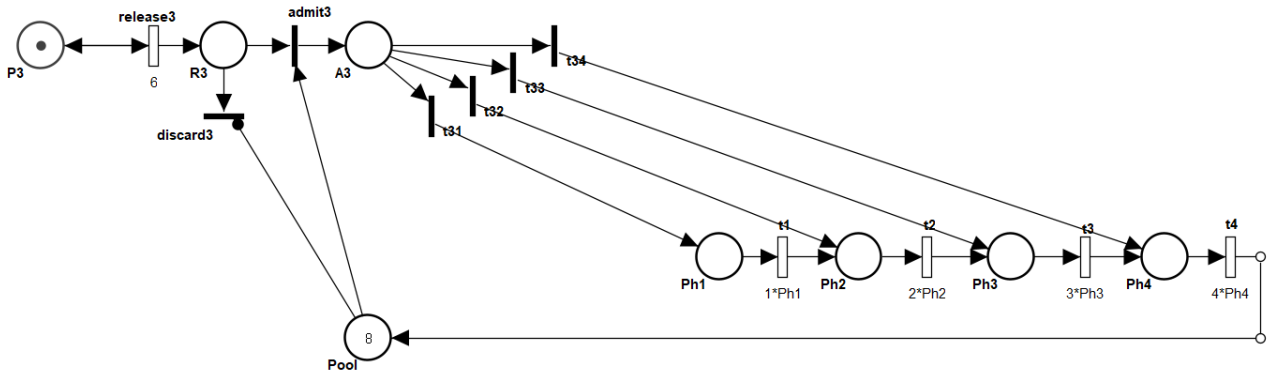
Spostando il gettone iniziale si possono studiare 3 diverse CDF, il modello permette di vedere come varia la CDF del BPH con gli weights delle transizioni di switch

Riportati qui di seguito a sinistra A1-1,1,1,1, al centro A2-1,2,3,4, a destra A3-4,3,2,1. I rewards nei plot plot sono Blu=Ph1 Verde=Ph2 Giallo=Ph3 Ciano=Ph4 Rosso=Empty. Nota che in A1 le probabilità di PH1,Ph2,Ph3, Ph4 si sovrappongono.



3.2 MODELLO SINGLE-CLASS

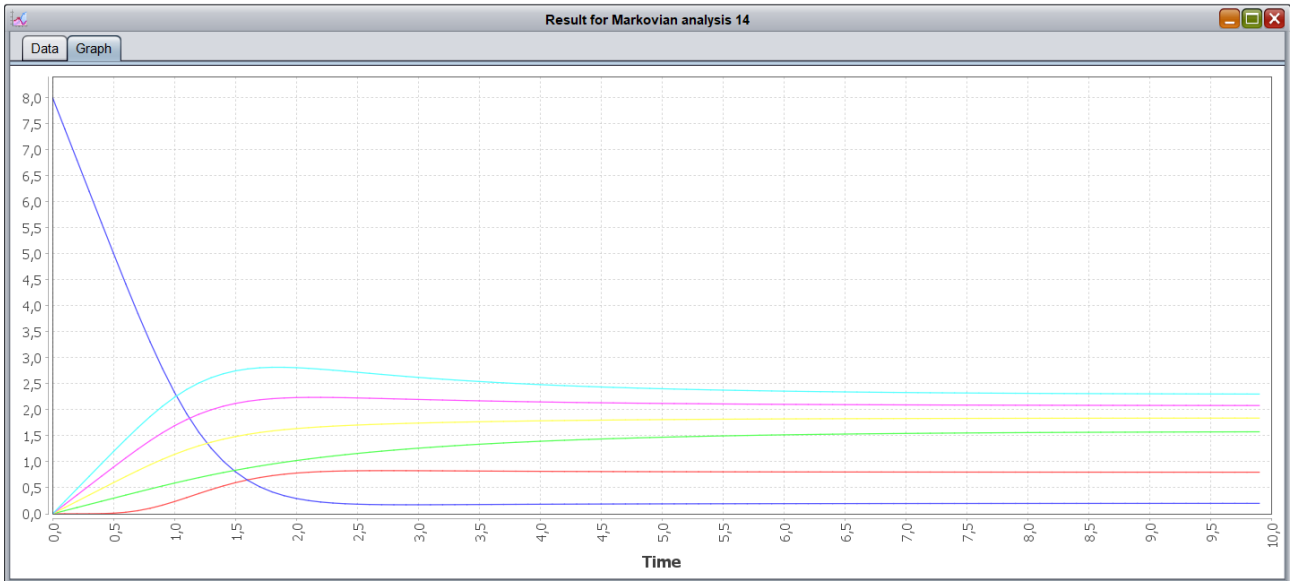
(parallelServerBernsteinPh_singleClass.xpn)



C'e' un processo di arrivo di Poisson (P3 e release3), e un pool parallel server con capacity 8 (Pool). Ogni istanza in servizio ha durata distribuita con un BPH individuato dalle probabilità di switch delle transizioni t31,t32,tr33,t34 che determinano le probabilità di ingresso nelle diverse fasi. I rilasci del processo di arrivo sono scartati se il Pool è esaurito (discard3) e ammesso se è disponibile un token da impegnare (admit3).

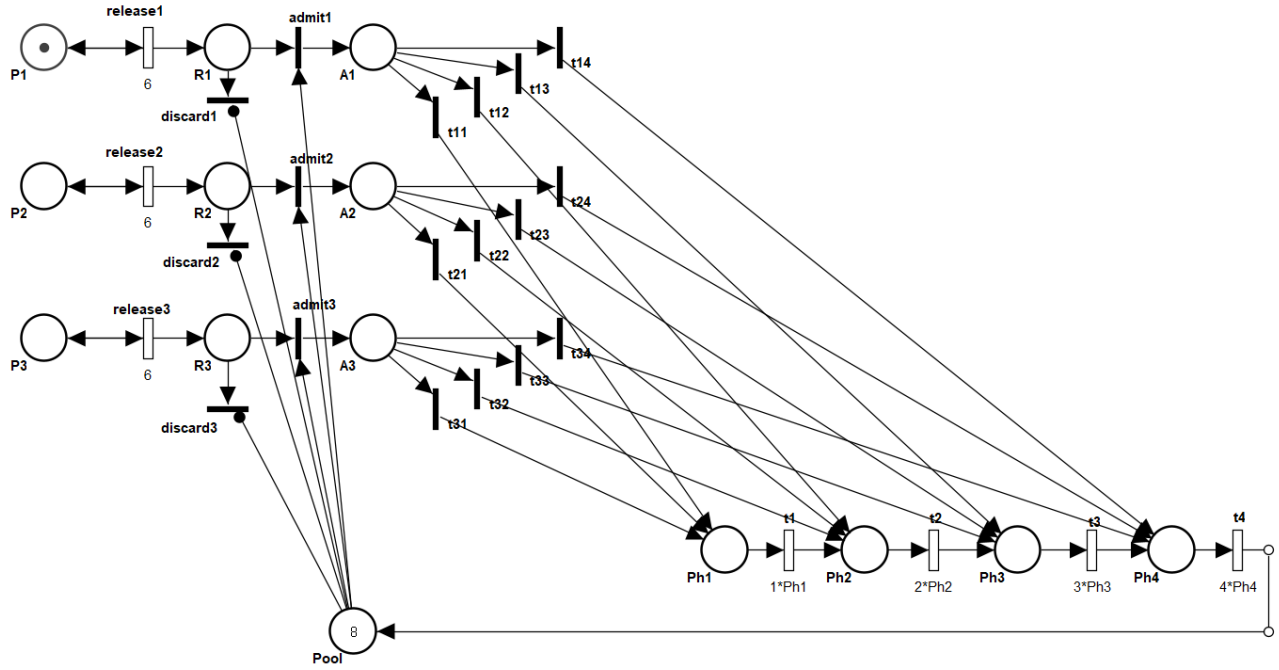
Il blocco Ph1,Ph2,Ph3,Ph4 con le transizioni t1,t2,t3,t4 rappresenta la durata del tempo di servizio, con distribuzione BPh di ordine 4 e probabilità di ingresso determinate dal random switch fra t31,t32,t33,t34.

L'analisi produce 3631 stati e si svolge in tempo molto breve (fino al tempo 4000 con passo 50) (Ph1=celeste, Ph2=ciano, Ph3=giallo, Ph4=verde, Pool, If(Pool==0,1,0)=blu)



3.3 MODELLO MULTI-CLASS

(parallelServerBernsteinPh.xpn)



Ci sono 3 classi di tasks, ciascuno con un suo processo di Poisson P1, P2 e P3, in questo caso con lo stesso tasso di arrivo 0.01.

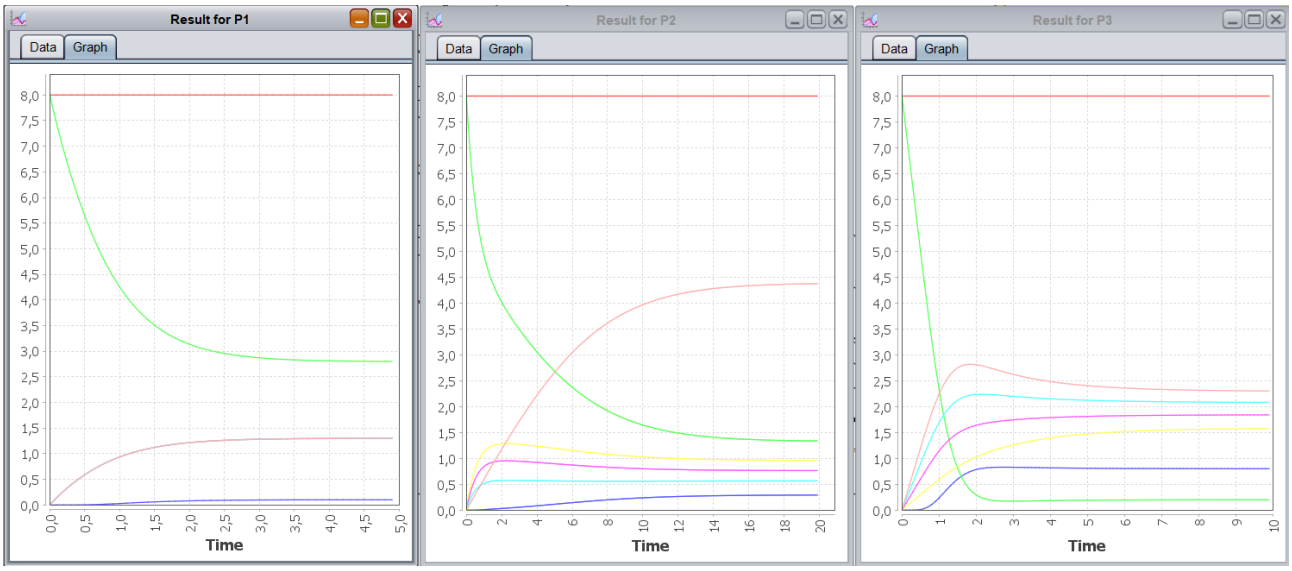
Ciascun processo ha una diversa distribuzione della durata del tempo di servizio, codificata attraverso i random switches determinati dagli weights delle transizioni tx1,tx2,tx3,tx4 che determinano con quale probabilità i tasks del processo x partono dalla fase Ph1,Ph2,Ph3 o Ph4.

Nel blocco in alto (P1) i pesi sono 1,1,1,1, in quello intermedio (P2) sono 1,2,3,4, in quello in basso (P3) sono 4,3,2,1. I tre casi hanno diverso valor medio, dove 1,2,3,4 ha valore medio piu' basso, e 4,3,2,1 il più alto. Le distribuzioni corrispondenti erano già discusse in un passo precedente di questo documento.

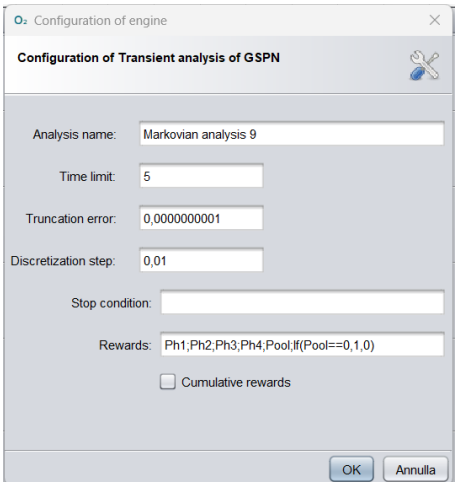
Spostando il gettone da P3 a P2 o P1 si può osservare il diverso comportamento di tre diversi sistemi con lo stesso processo di arrivo ma diversa distribuzione del tempo di servizio. L'analisi si fa bene con transient GSPN, che produce in ciascun caso 3631 stati.

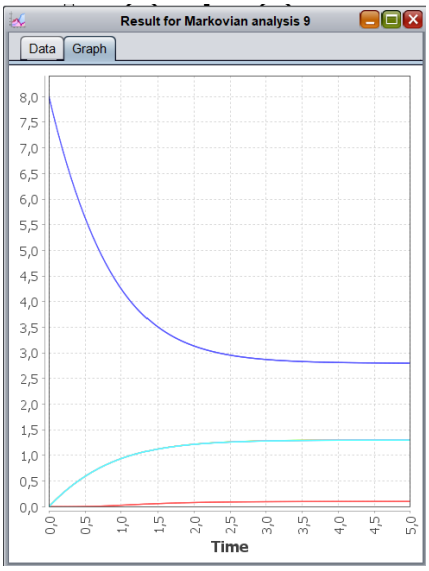
Lo steady state è raggiunto con tempi diversi, e il valore medio di tokens nel pool si assesta su valori diversi. Come prevedibile P2 ha un carico "più leggero" di P3 perché sposta "più avanti" la fase di ingresso nel BPH. Questo si riflette bene sul diverso plot del numero di tokens nel pool (verde) e nel diverso valore medio di Pool pieno (rosso).

Analizzando con errore di troncamento 0,00000001 (+00) si ottiene per i 3 casi: (verde=Pool, rosso forte un check che deve fare somma 8, rosso chiaro Ph1, celeste Ph2, ciano Ph3, giallo Ph4, blu Pool==0)



NON SI CAPISCE PERCHE' l'andamento del caso P1 non è un intermedio fra P2 e P3, e sembra invece comportarsi meglio di P2. Non dovrebbe essere un errore numerico: provando a ridurre l'errore di troncamento ulteriormente e analizzando con passo 0.01, … ma l'esito non varia:

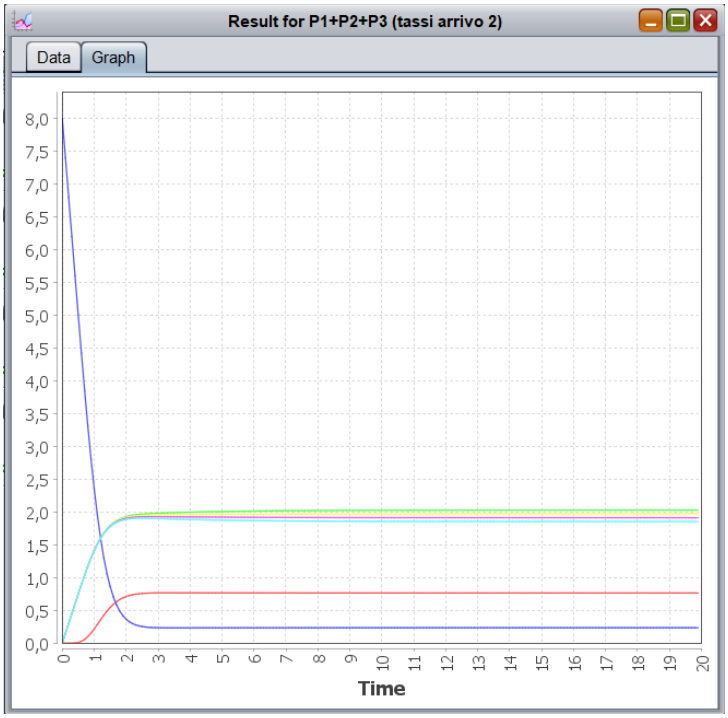




Nemmeno simulando con il token game si vedono effetti strani. Si osserva solo che è molto forte l'impatto dei rates crescenti, per cui un token in Ph1 tende a restare lì, distanziato da quelli che

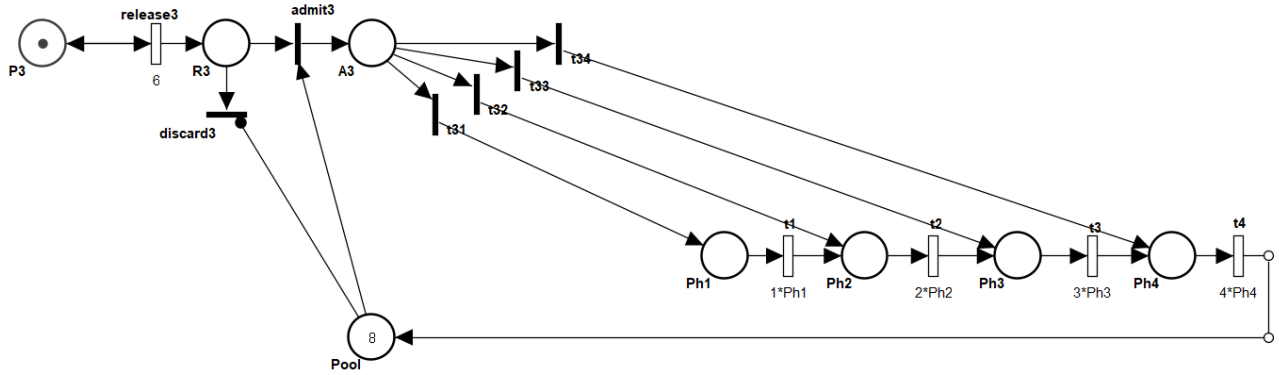
anche se entrano dopo vanno pero' a finire in fasi piu' avanzati e procedono piu' rapidamente verso il completamento … che suggerisce che la distribuzione piu' interessante da studiare è 4,3,2,1, i.e. il P3.

Sommando i carichi (i.e. mettendo un gettono in P1, P2,e P3 contemporaneamente) e riducendoli per la stabilità (i.e. tassi di arrivo 2 anziché 6 per tutti e tre i processi), la complessità non varia che cis sono cmq lo stesso numero di stati, si ottiene:



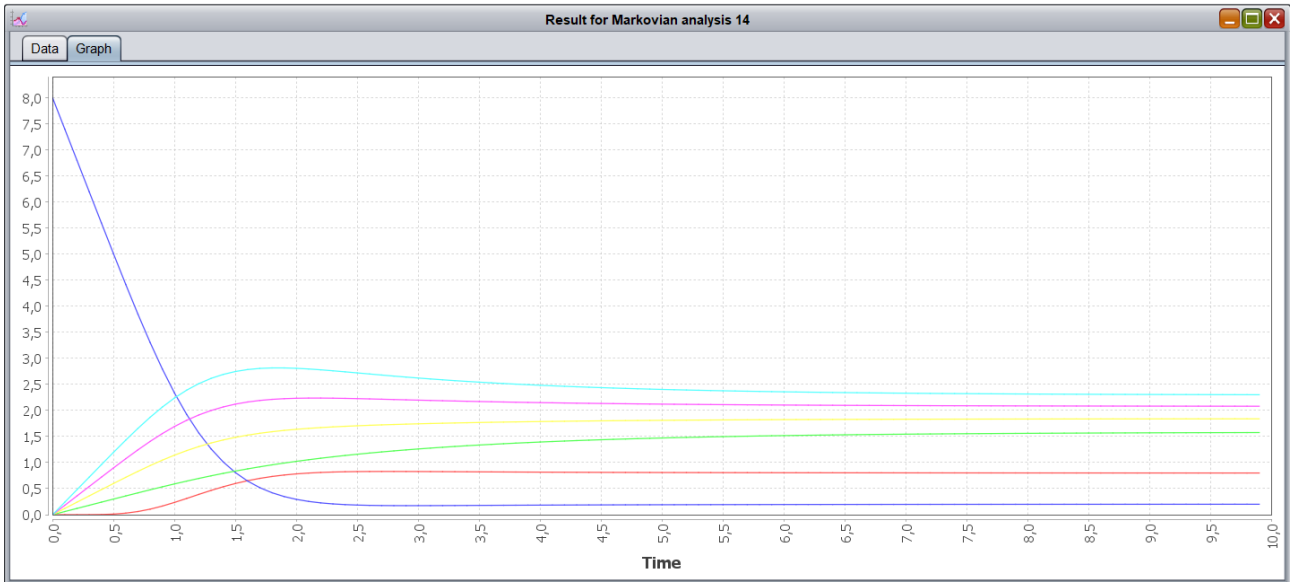
3.4 MODELLO SINGLE CLASS CON 8 FASI

(parallelServerBernsteinPh_singleClass.xpn)



Torniamo al caso single class, che evita di combinare processi di impatto diverso o diverso settling time. In particolare sperimentiamo con P3, che ha la distribuzione piu' interessante delle fasi di ingresso nel BPH (4,3,2,1).

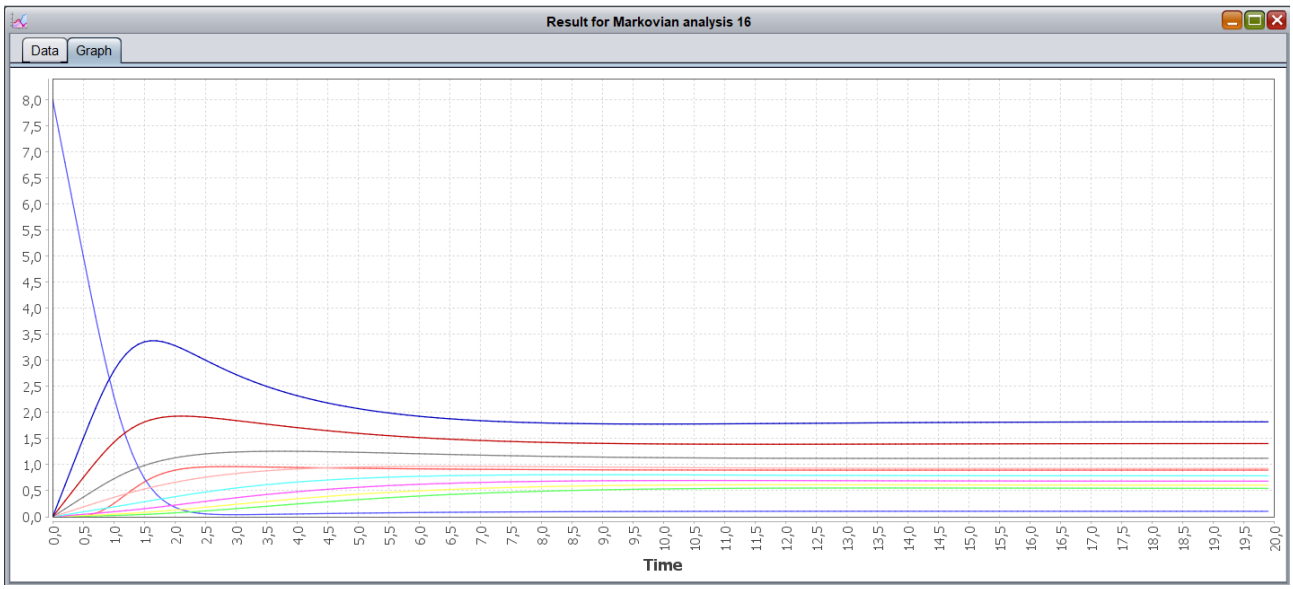
L'obiettivo è ora sperimentare come varia la feasibility dell'approccio con il numero di fasi nel BPh.



Con 8 fasi escono fuori 128701 stati, l'analisi fino a 20 con passo 0,1 si conclude cmq in qualche decina di secondi.

Azzurro chiaro Pool (nota che va circa a 0, il ssitema è troppo caricato, il tasso è 6 come era pe ril caso a 4 fasi 4,3,2,1, qui ci sono 8 fasi e con peso molto piu' sbilanciato verso il basso 128,64,32,16,8,4,2,1); poi a discendere le fasi nel loro ordine Blu=Ph1 rosso=Ph2 grigio=Ph3, rosa =Ph4, celeste=Ph5, ciano =Ph6, giallo=Ph7, verde=Ph8.

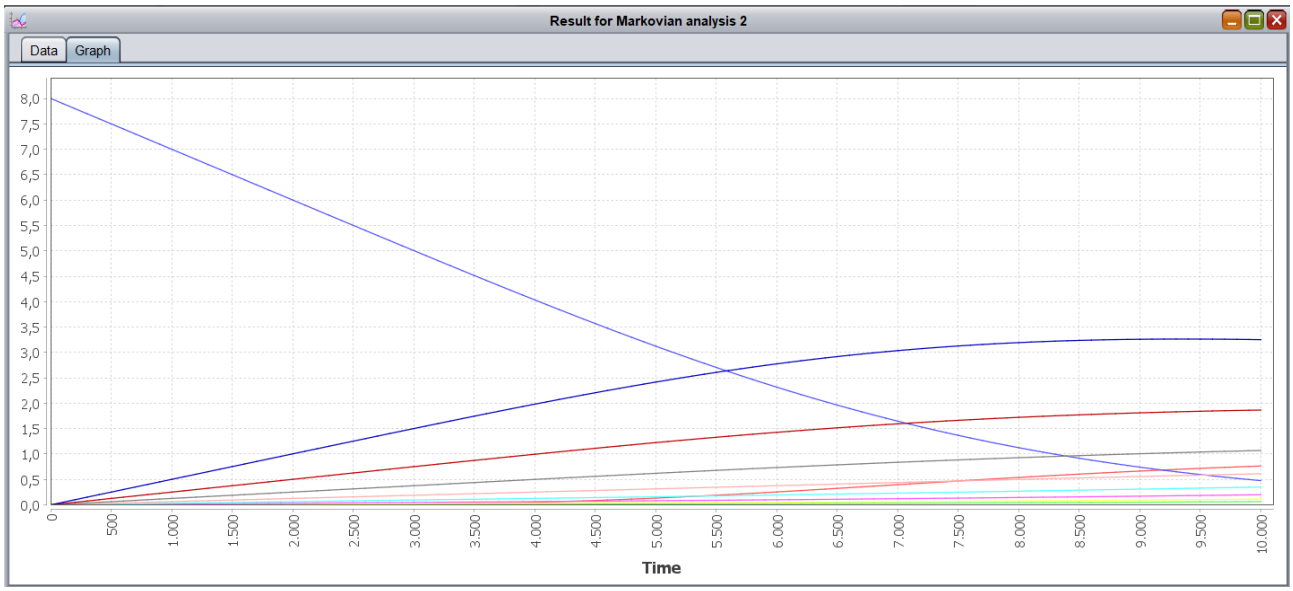
Il rosso vermiglio è il Pool pieno (i.e. If(Pool==1,0,1)), che va circa a 1, i.e. il Pool è quasi sempre pieno, come già osservato il sistema è sovraccaricato.



Riduciamo il carico (l'alternativa sarebbe aumentare il pool … ma poi la complessità espolode davvero). Il problema è che il carico va ridotto parecchio, e quini il transiente si allunga molto.

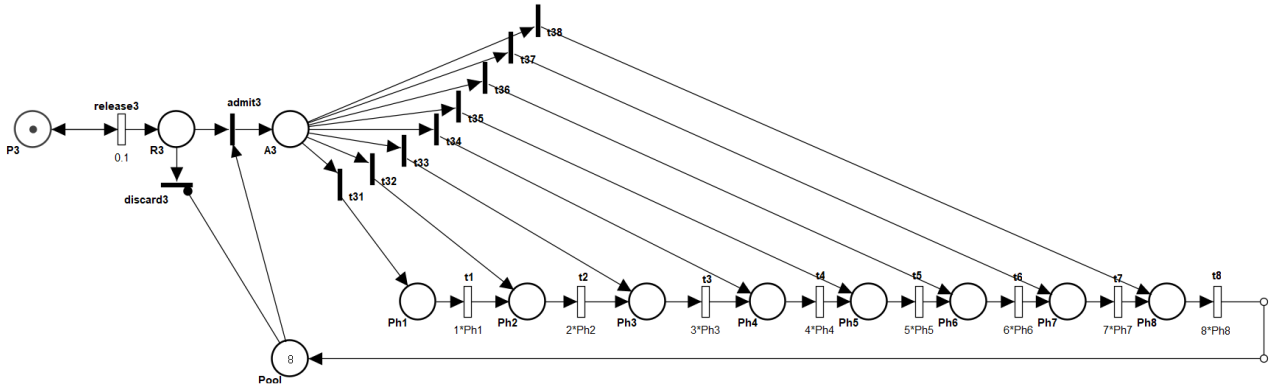
(E' un peccato non potere usare l'analisi steady state!)

Con tasso di arrivo 0.001, si ottiene

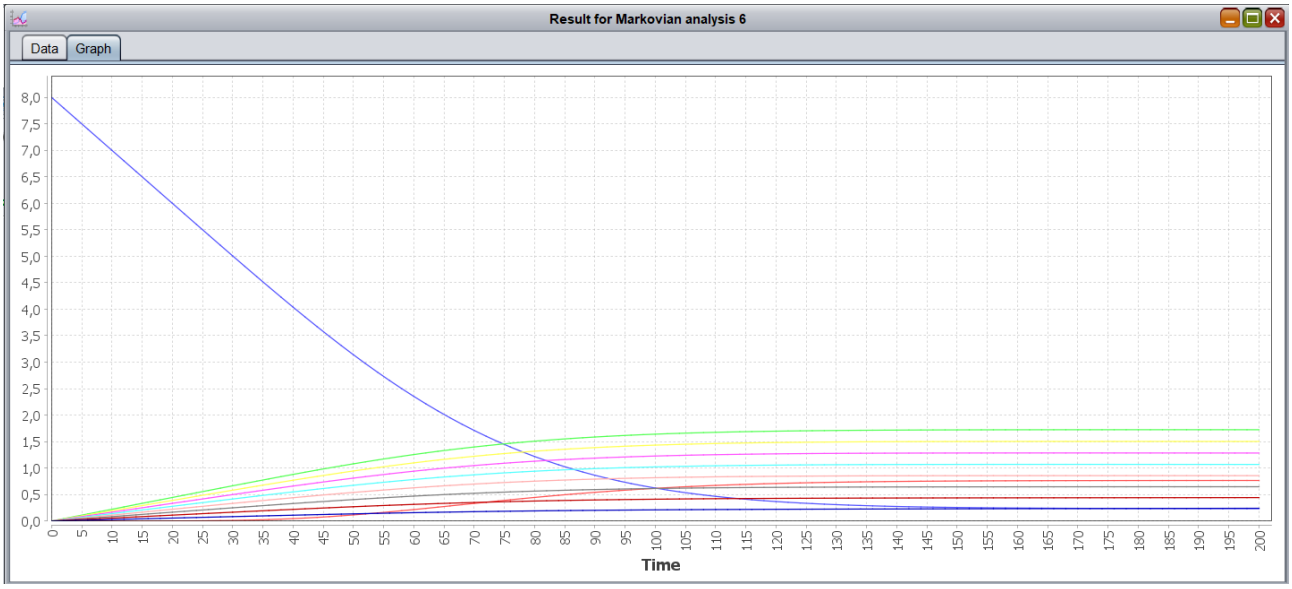


Con 8 fasi si fa molta fatica, non tanto per l'analisi, che sarebbe sopportabile, ma piuttosto per l'ampiezza del settling time … e probabilmente anche per la distanza nell'ordine di grandezza fra tempo medio fra arrivi (va portato fino a 1000) e tempi di avanzamento delle fasi (che sono cmq dell'ordine di 1,1/2,1/4 … 1/128).

Possiamo ridurre il carico modificando il tempo di servizio, che ora era davvero molto sbilanciato. Passiamo allora a campioni in forma aritmetica con probabilità di ingresso nelle fasi 1,2,3,4 … … ancora troppo carico. E troppo carico anche la versione centrata 1,2,3,4,4,3,2,1. Quindi alal fine provo con l'ordien inverso 1,2,3,4,5,6,7,8 (con prob max di entrare nell'ultima fase). Con questo si trova un punto ragionevole con tasso di arrivo 0.1:

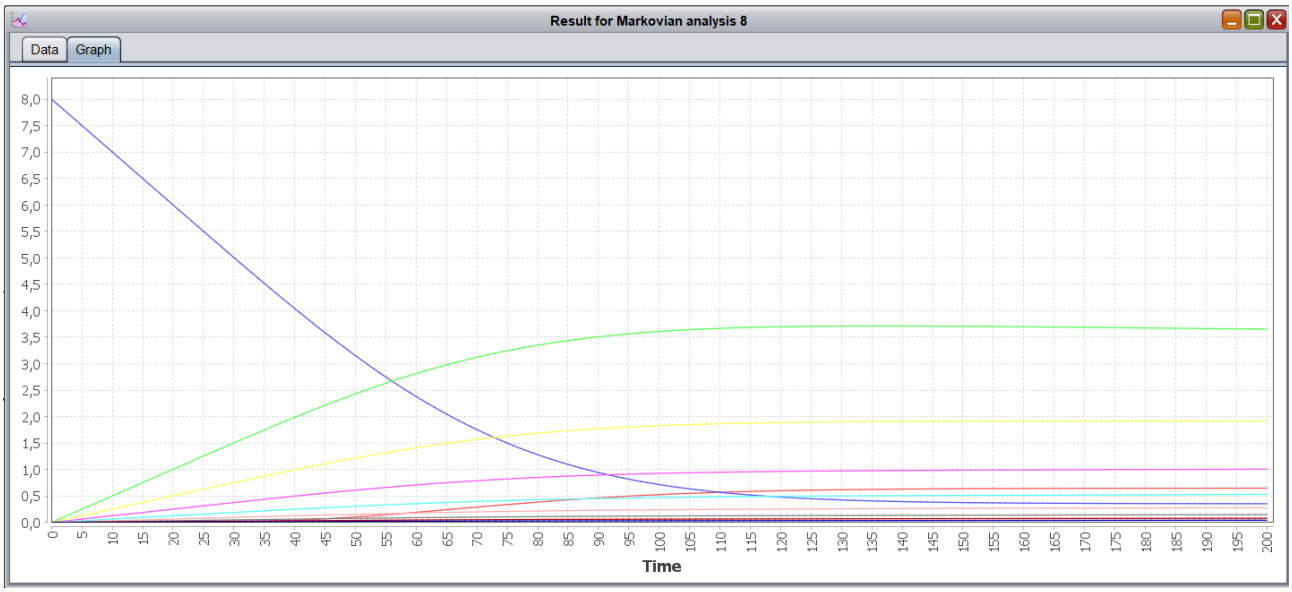


Analisi fino a 200 con passo 2:

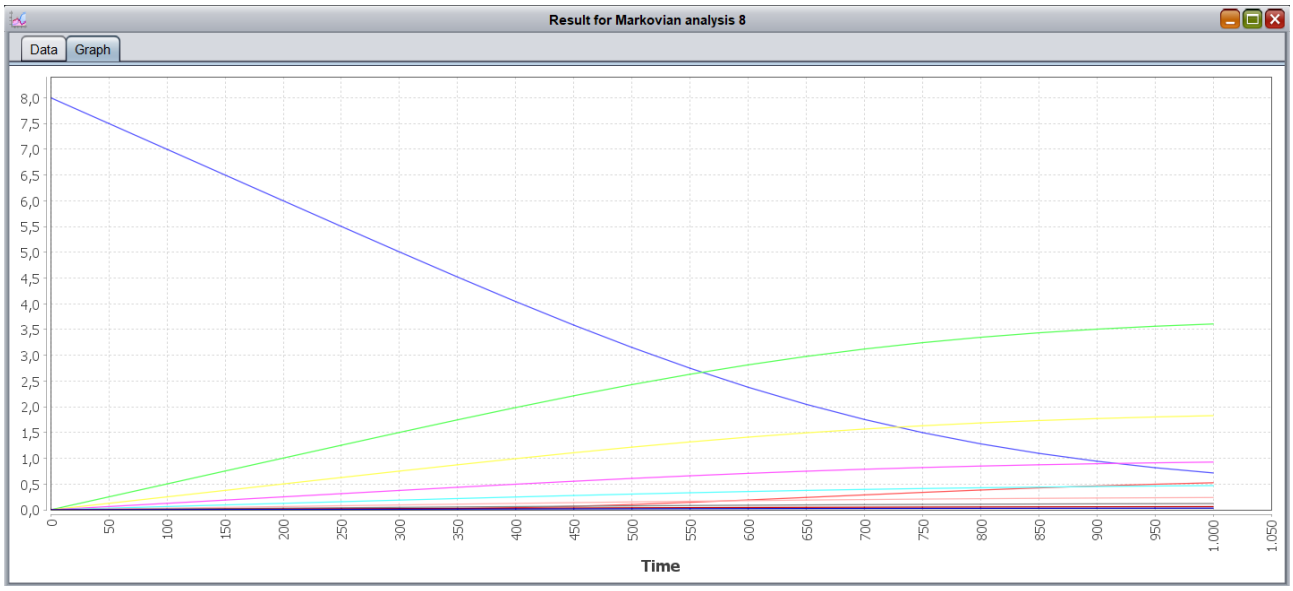


La prob di avere il Pool vuoto va cmq a 0.766 …

Quindi riduco ancora il tempo di servizio, passando alla sequenza 1,2,4,8,16,32,64,128 … sempre con tasso di arrivo 0.1 … va un po' meglio, ma cmq nitidamente sovraccarico:



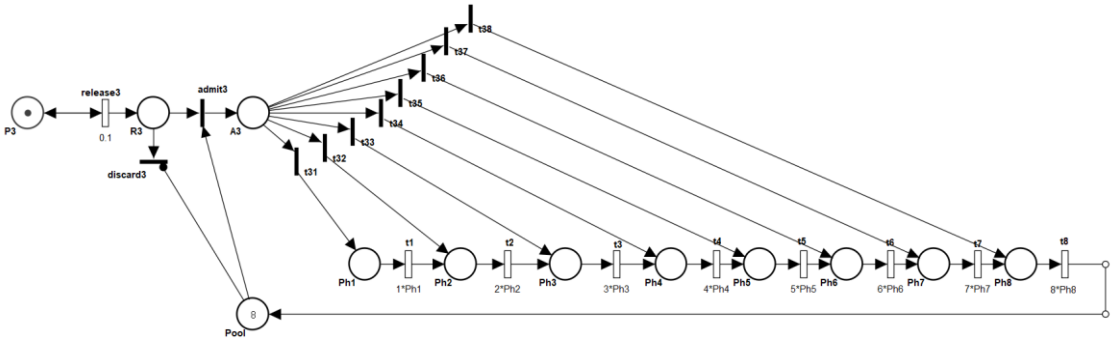
Portando il tasso di arrivo a 0.01 si ottiene una performance più accettabile, ma stiamo molto stressando la stiffness del problema: tempo emdio fra arrivi 100 e tempo medio di avanzamento nell'ultima fase 1/8 (o anche emno se c'e' piu' di un gettone):



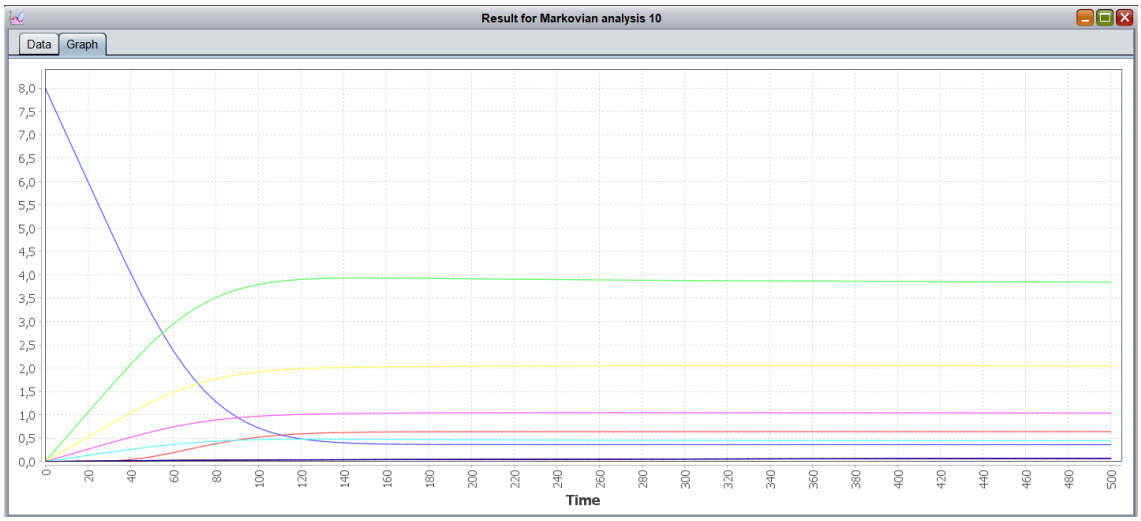
A occhio, conviene riconoscere che 8 fasi portano necessariamente a stiffness … a meno di non avere un profilo di carico che mette quasi tutto sulle ultime fasi.

E.g. con 1,1,1,1,16,32,64,128 e tasso di arrivo 0.1:

(parallelServerBernsteinPh_singleClass8Phases1-1-1-1-8-16-32-64Light.xpn)



Otteniamo:



Cmq, si capisce che 8 fasi sono tante, e che forse è meglio lavorare con 6.

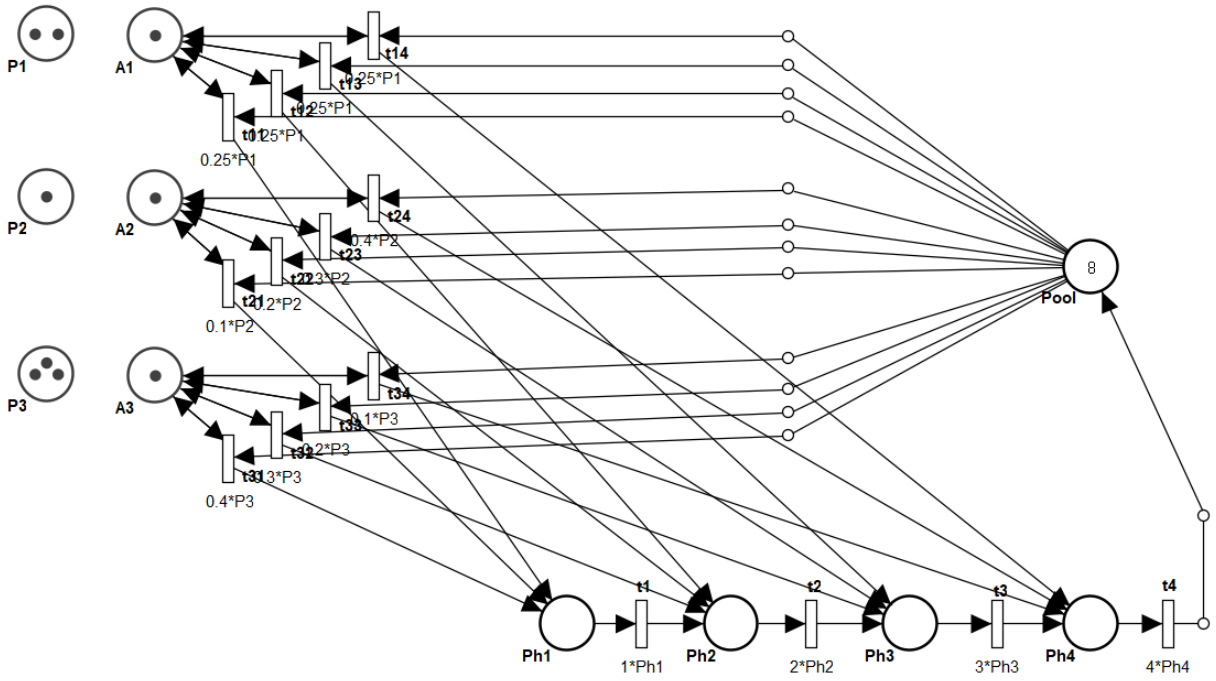
4 MODELLO IMM-FREE

Notoriamonete Oris ha un bug sul trattamento di GSPN con EXP e IMM … prima o poi lo correggiamo, nel frattempo facciamo meglio a lavorare su un modello in cui abbiamo rimosso le IMM in modo che

4.1 4 FASI

sia equivalente nella misura sul tempo-continuo

(parallelServerBernsteinPh_NoImm.xpn)

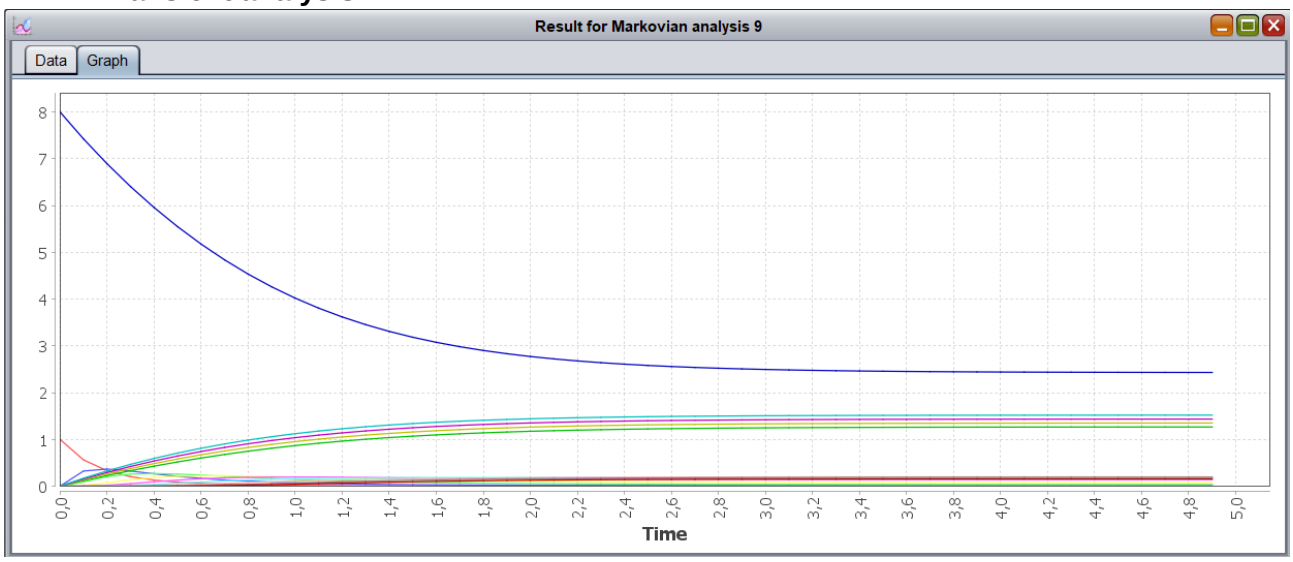


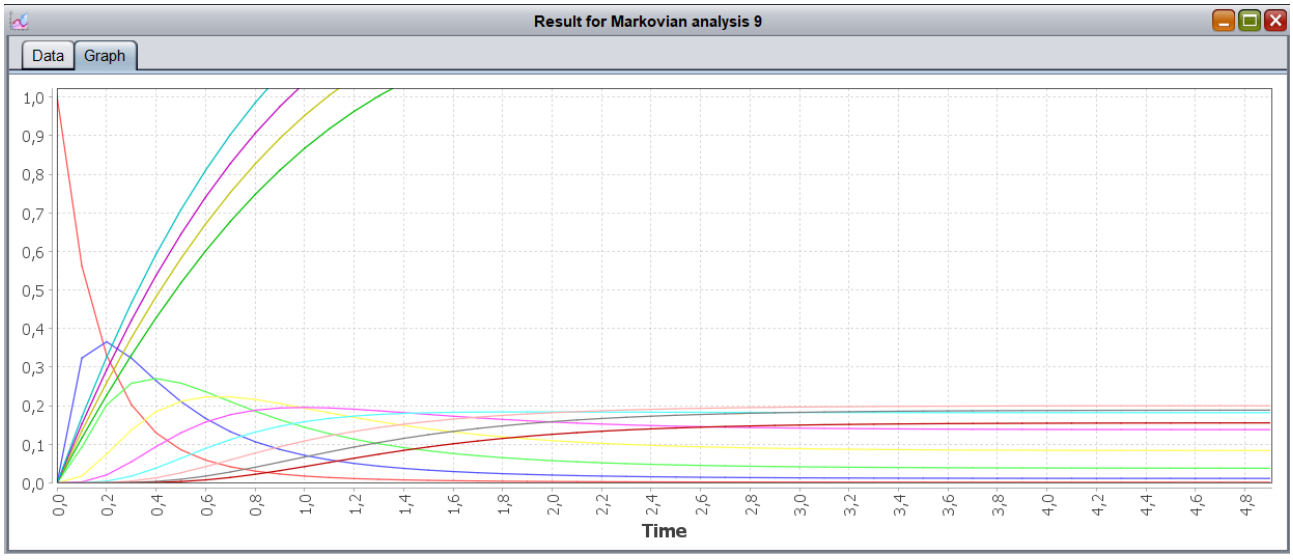
4.1.1 Analisi steady state

If(Pool==0,1,0)	0.15582582254977684
If(Pool==1,1,0)	0.18887978490882426
If(Pool==2,1,0)	0.20032704460026812
If(Pool==3,1,0)	0.18211549509115282
If(Pool==4,1,0)	0.1379662841599641
If(Pool==5,1,0)	0.08361592979391755
If(Pool==6,1,0)	0.03800724081541701
If(Pool==7,1,0)	0.011517345701641498
If(Pool==8,1,0)	0.001745052379036589
Ph1	1.519513519410362
Ph2	1.435096101665374

Ph3	1.3506786839203646
Ph4	1.2662612661753416
Pool	2.4284504288285484

4.1.2 Transient analysis





5 E QUINDI, COSA FARE

Analisi di sensitività rispetto ai parametri, tassi di arrivo, distribuzione dei tempi di servizio (i.e. tassi delle transizioni che fanno race-condition per decidere la fase di ingresso nel Phase Type), pool size,

… e rispetto a requisiti: tasso di rejection, numero di containers inattivi

Ricercare un punto ottimo di operazione,

analisi di elasticità: partendo dallo steady state, valutare il settling time con cui li sistema risponde ad una perturbazione. E.g Fallisce un Container, oppure il carico ha un picco

performability: capacità di mantenere performance in presenza di fallimenti (performance+reliability)

analisi del grado di adattabilità:

Studiare l avraiabilità del tasso di arrivo ?

Definire un apolitica di scaling delel risorse (Poolsize) basata sull'analisi tenendo conto della previsioen sul tasso di arrivo costruita su una previsione sul tasso di arrivo

Prendiamoci dle tempo per definire lo scenario in cui finalizziamo la caapcità di analisi.

Pe rintanto i passi sono:

sperimentare con Oris e con i modelli che abbiamo visto … e poi con Sirio … si arriva fino ad avere un impianto che permetet di studiare il modello (sensitivity analysis, elasticicty …)

poi valutimao in base alal difficoltà incontrata se diventa possibile sperimentare tecniche di adattamento dle poolsize.