Aula 7 - Trabalho prático e Algoritmo Hi-DSD

Monday, March 28, 2016 1:35 PM

Especificação do trabalho prático 1

Implementar, usando SMPL, um algoritmo de diagnóstico adaptativo distribuído em que os nodos, ordenados sequencialmente de 0..N-1, a cada intervalo de testes, escolhem um nodo aleatório para tes star (distribuição uniforme). Se o nodo testado estiver falho, o testador continua* até encontrar um nodo sen n-falha, do qual obtém informações de diagnóstico sobre todos os demais nodos - exceto aqueles que testou. Se tes star todos os nodos falhos, dorme para iniciar os testes novamente no próximo intervalo.
*continua testando nodos aleatórios, excetuando aqueles que já testou e si mesmo e paran do após testar todos os outros nodos.

IMPORTANTE: Medir a latência para a detecção de 1 (um) evento. Após a ocorrência de um evento, medir quantas rodadas de testes são necessárias para todos os nodos completarem o diagnóstico.
Contar também o número de testes executados, desde a ocorrência do evento até completa ar o diagnóstico.

OBS: Um evento corresponde à falha de um ou mais nodos.

Deve haver um modo de execução denominado "síntese" em que, automaticamente, são ex ecutadas 30 simulações para 30 eventos aleatórios. Nesse caso, mostrar a média e o desvio padrão.ções de todos os outros nodos

Como mergir vetor de estados

Se o nodo i testar os nodos j e k, ele recebe informações de todos os outros nodos.

O algoritmo Hi-ADSD

A latência do algoritmo Adaptive-DSD é muito alta: N rodadas de testes no pior caso.

Por exemplo: considere uma LAN com N=100 hosts
Intervalo de de testes = 30 segundos
Latência: $30s \times 100 = 3000s = 50min$

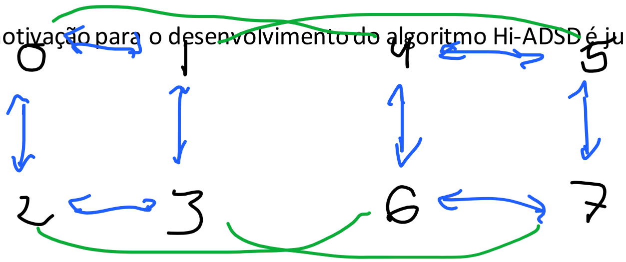
Soluções para diminuir o tempo:

1. "Event-driven diagnosis"
        Quando um evento é detectado, dispara disseminação desta informação.
        BROADCAST confiável (*reliable broadcast*). Duas alternativas:

BROADCAST confiável (*reliable broadcast*). Duas alternativas:

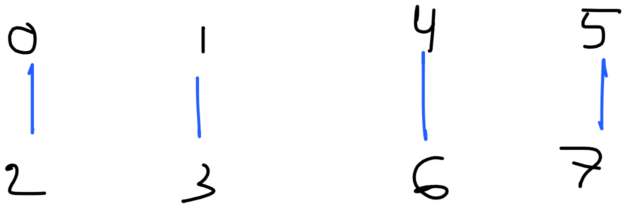
- Algoritmo da força bruta: **muito caro**.
- Algoritmo "barato": usa um detector de falhas (aka usa as informações de diagnóstico! ciclo vicioso)

A motivação para o desenvolvimento do algoritmo Hi-ADSD é justamente reduzir a latência do próprio diagnóstico.

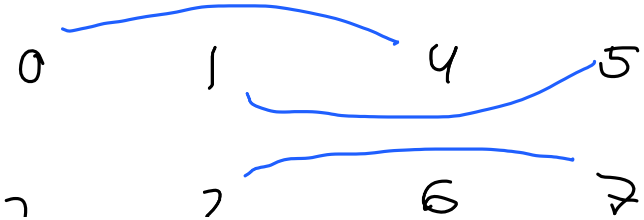


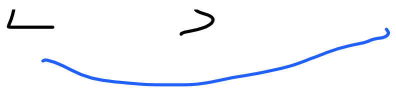
Primeira rodada de testes:

Segunda rodada de testes:

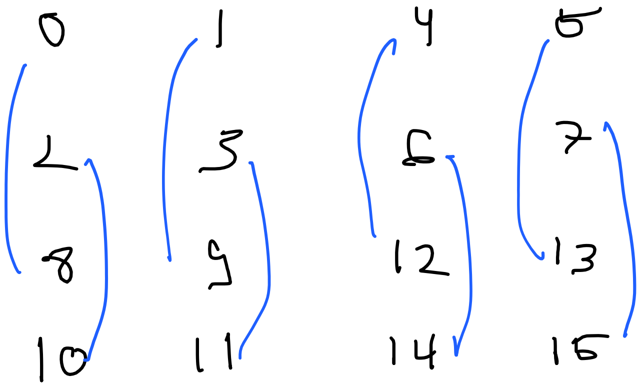


Terceira rodada de testes

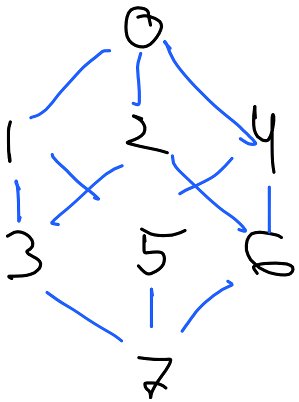


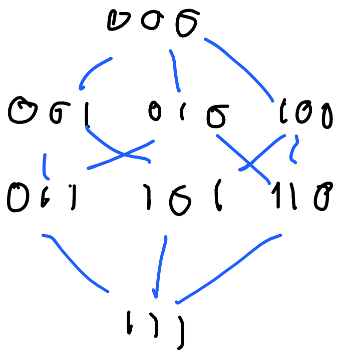


Na quarta rodada de testes, com 16 nodos



O grafo de testes do algoritmo Hi-ADSD, quando todos os nodos estão sem-falhas, é u um hiper-cubo





No hipercubo de dimensão d há 2^d nodos. Cada nodo tem um identificador de d bit s. Só existe uma aresta entre dois nodos se seus identificadores diferem em um único bit.

No algoritmo Hi-ADSD os nodos executam testes em "clusters".

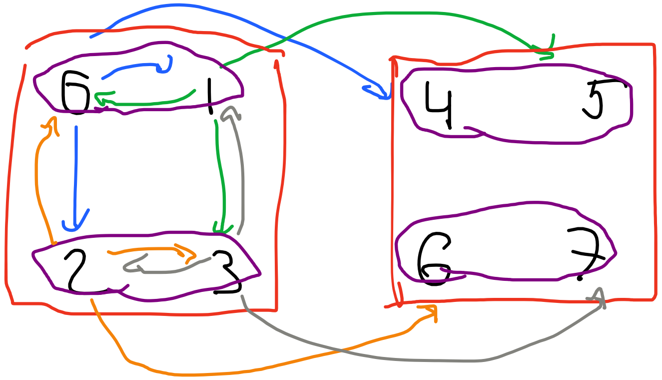
- Um cluster é ~~um conjunto~~ uma lista de nodos
- Em cada intervalo de testes s um nodo i testa o cluster $C_{i,s}$ até encontrar um nodo sem-falha ou testar todos os nodos do cluster falhos; se encontrar um nodo sem-falha, obt ém info de diagnóstico do cluster a parter daquele nodo (exceto os testes que realizou).

Exemplo: $N = 8; d = 3$
No primeiro intervalo de testes, o nodo i testa o cluster $C_{i,1}$.
No segundo intervalo de testes, o nodo i testa o cluster $C_{i,2}$.
No terceiro intervalo de testes, o nodo i testa o cluster $C_{i,3} = C_{i,d}$.
No quarto intervalo de testes, o nodo i testa o cluster $C_{i,1}$.
E assim por diante.

A tabela a seguir mostra, ∀ i, $0 \leq i \leq 7$, os valores de $C_{i,1} \dots C_{i,d=3}$

s	$C_{0,s}$	$C_{1,s}$	$C_{2,s}$	$C_{3,s}$	$C_{4,s}$	$C_{5,s}$	$C_{6,s}$	$C_{7,s}$
1	1	0	3	2	5	4	7	6
2	2, 3	3,2	0,1	1,0	6,7	7,6	4,5	5,4
3	4,5,6,7	5,6,7,4	6,7,4,5	7,4,5,6	0,1,2,3	1,2,3,0	2,3,0,1	3,0,1,2

Testes são realizados em sequência caso nodos sejam falhos. i.e. se 0 testar 2 como falho na segunda rodada, ele testa 3.



Considerando um sistema S de N nodos $0..N-1$, um cluster de K nodos $n_j..n_{j+k}$ $_{-1}$ é tal que $j \bmod k = 0$, k potência de 2; o cluster é definido como n_j se $k = 1$ ou a união de dois clusters $n_j..n_{j+\frac{k}{2}-1}$ e $n_{j+\frac{k}{2}}..n_{j+k-1}$.