

GAME OF --- NODES

FEATURING



ALEXANDRE
FRUCHAUD

DELPHIN AUBIN



“Winter is coming”

“Avant on utilisait
Oracle”

“Winter is coming”

MASTER & SLAVE



MASTER & SLAVE



Si le master tombe un slave
prend le relais

MASTER & SLAVE



Si le master tombe un slave
prend le relais

Tous les slave peuvent lire

MASTER & SLAVE



Si le master tombe un slave
prend le relais

Tous les slave peuvent lire

Le master doit attendre
tous les slave

“Winter is coming”



NETFLIX

“Winter is coming”

amazon

Google

facebook





NOSQL



CAP

AVAILABILITY



CONSISTENCY

PARTITION TOLERANCE



AVAILABILITY

“Toutes les requêtes
reçoivent une réponse”

CONSISTENCY

“Chaque lecture lit la dernière écriture ”



PARTITION TOLERANCE

“Aucune panne ne doit empêcher le système de répondre correctement”



AVAILABILITY



CONSISTENCY

PARTITION TOLERANCE

AVAILABILITY

ORACLE



CONSISTENCY

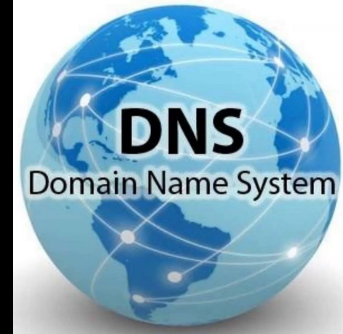
PARTITION TOLERANCE

AVAILABILITY

ORACLE



cassandra



CONSISTENCY

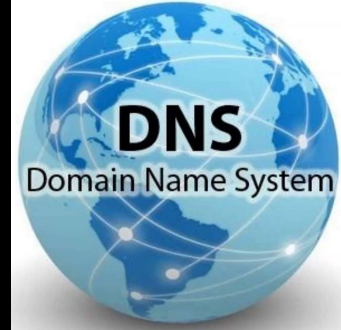
PARTITION TOLERANCE

AVAILABILITY

ORACLE



cassandra



Apache ZooKeeper™ RANCE

“L’avantage avec CAP c’est surtout
que ça te permet de justifier
pourquoi tu ne fais pas la 3ème
lettre”



AVAILABILITY

“C’est facile d’être
d’accord avec soit
même”

CONSISTENCY

PARTITION TOLERANCE

AVAILABILITY

“C’est facile d’être
d’accord avec soit
même”

“C’est facile de
dire n’importe
quoi à tout le
monde”

CONSISTENCY

PARTITION TOLERANCE

AVAILABILITY

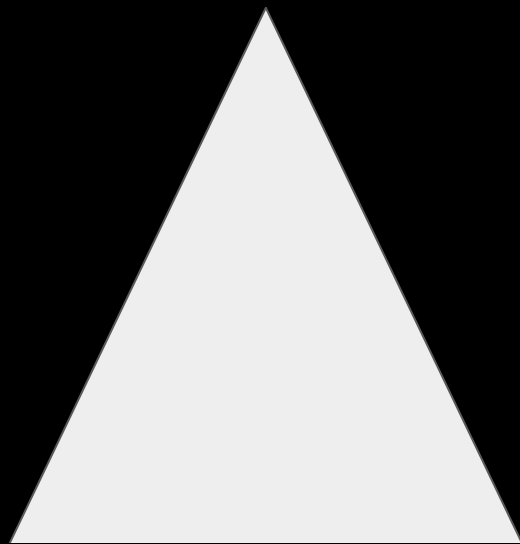
“C’est facile d’être
d’accord avec soit
même”

“C’est facile de
dire n’importe
quoi à tout le
monde”

CONSISTENCY

PARTITION TOLERANCE

“Mettre tout le monde d’accord c’est plus dur”



RAFT

Let's simplify !

Le log est append only

1	3	7	9	12	27	5	8	...
---	---	---	---	----	----	---	---	-----



Temps

GOTCODE?

```
previousElements = [];  
selectedScopes = [];  
  
scope.$watch(watchExpr, function ngSwitchWatchAction(value) {  
    var i, ii;  
    for (i = 0, ii = previousElements.length; i < ii; ++i) {  
        previousElements[i].remove();  
    }  
    previousElements.length = 0;  
  
    for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
        selectedElements[i].remove();  
        selectedScopes[i].destroy();  
        previousElements[i] = selected;  
        scope.$eval(selected.$watchExpr, function ngSwitchAction() {  
            previousElements.splice(i, 1);  
        });  
    }  
  
    selectedElements.length = 0;  
    selectedScopes.length = 0;  
  
    if ((selectedTranscludes = ngSwitchController.cases['!' + value] || ngSwitchController.defaultCase)) {  
        scope.$eval(attr.change);  
        forEach(selectedTranscludes, function(selectedTransclude) {  
            var selectedScope = scope.$new();  
            selectedScopes.push(selectedScope);  
            selectedScope.$parent = scope;  
            selectedScope.$watch(selectedTransclude.$watchExpr, function ngSwitchTranscludeWatchAction() {  
                selectedScope.$eval(selectedTransclude.$watchExpr);  
            });  
            selectedTransclude.$transclude(selectedScope, function(clone, scope) {  
                clone.appendTo(selectedElements);  
            });  
        });  
    }  
});
```

```
selectedElements.length = 0;  
selectedScopes.length = 0;
```

```
if ((selectedTranscludes = ngSwitchController.cases['!' + value] || ngSwitchController.defaultCase)) {  
    scope.$eval(attr.change);  
    forEach(selectedTranscludes, function(selectedTransclude) {  
        var selectedScope = scope.$new();  
        selectedScopes.push(selectedScope);  
        selectedScope.$parent = scope;  
        selectedScope.$watch(selectedTransclude.$watchExpr, function ngSwitchTranscludeWatchAction() {  
            selectedScope.$eval(selectedTransclude.$watchExpr);  
        });  
        selectedTransclude.$transclude(selectedScope, function(clone, scope) {  
            clone.appendTo(selectedElements);  
        });  
    });  
}
```


“Le **leader** doit savoir si ses **follower** sont à jour avant de répondre ok”



“Le commit



“Le **commit** est le point du log



“Le **commit** est le point du log avant lequel toutes les entrées ont été **ack**”



“Le **commit** est le point du log avant lequel toutes les entrées ont été **ack** par la majorité des nodes.”





STARK

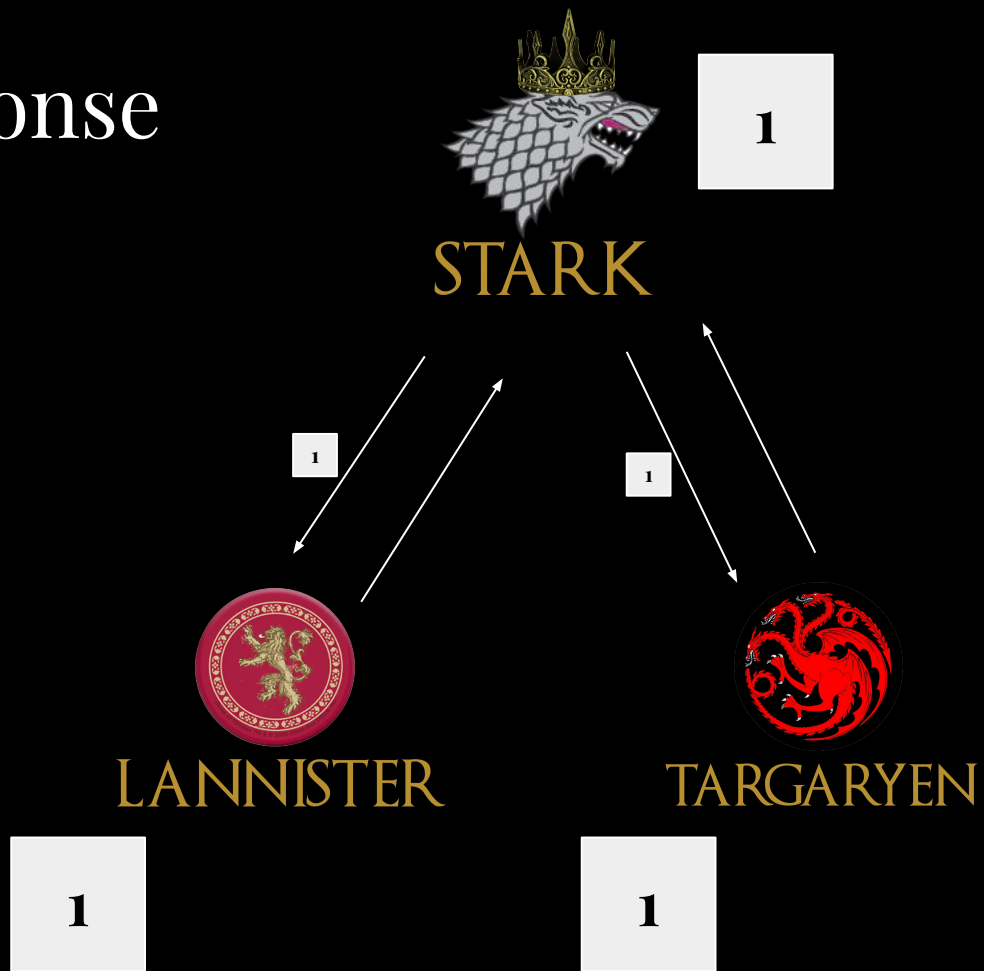


LANNISTER

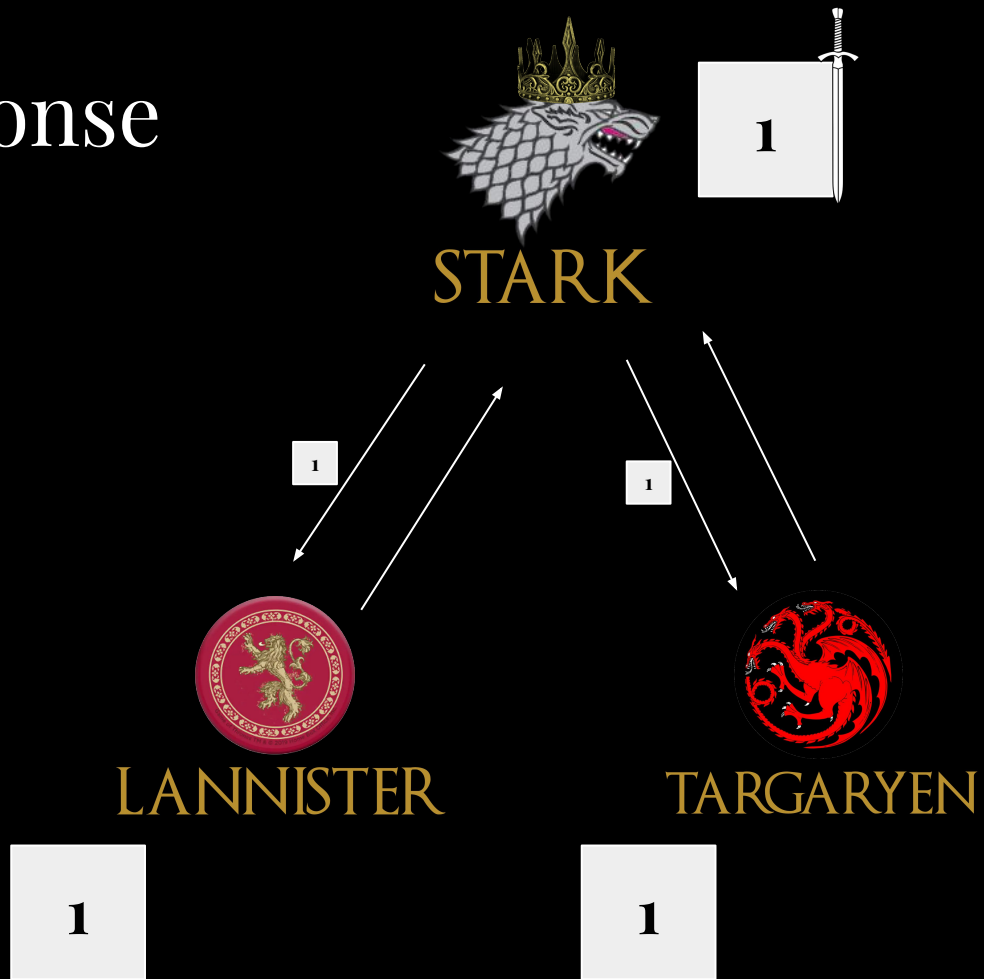


TARGARYEN

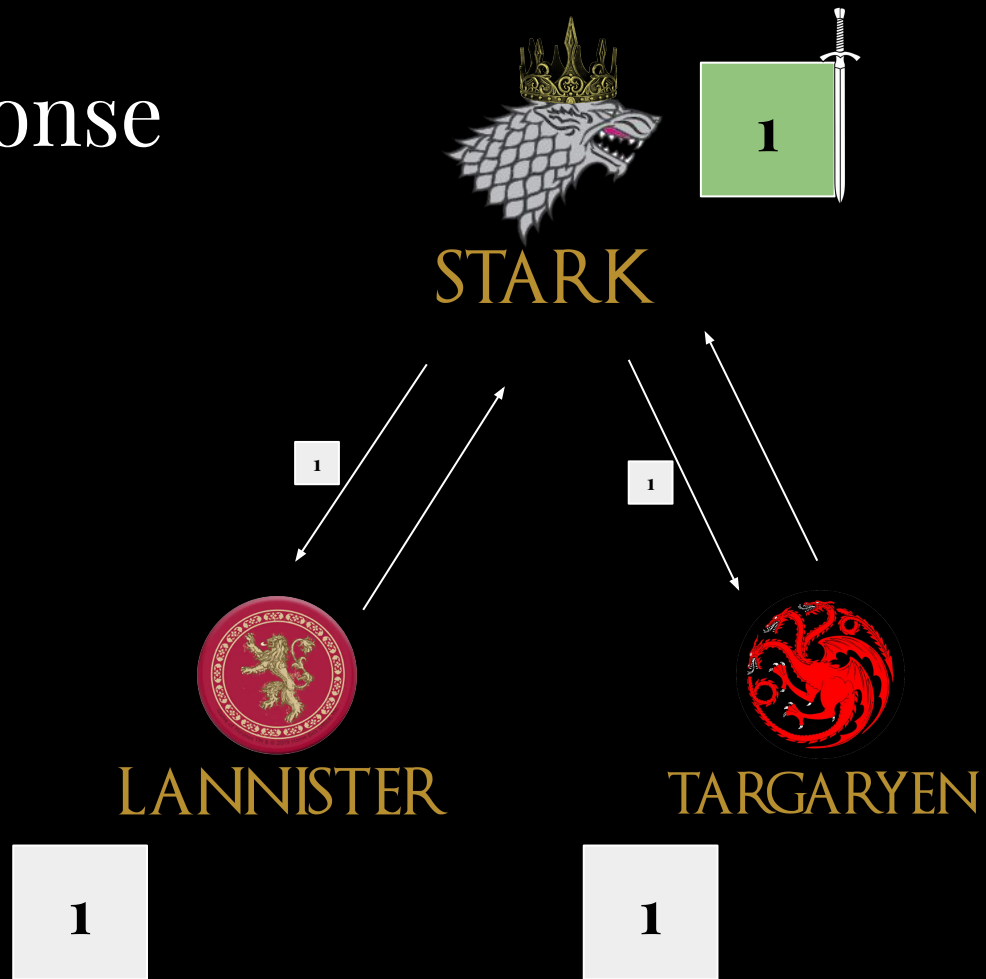
LogResponse

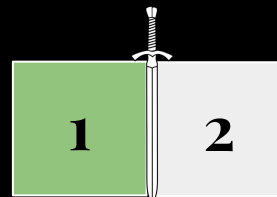


LogResponse



LogResponse





STARK

1



LANNISTER

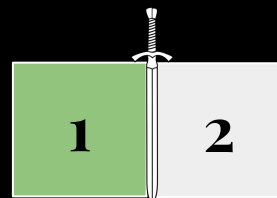
1



TARGARYEN

1

1



STARK

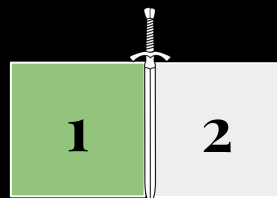


LANNISTER



TARGARYEN





STARK



LANNISTER

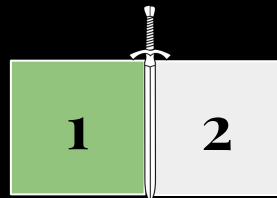


TARGARYEN





STARK



ackLength: 2

ackLength: 2



LANNISTER

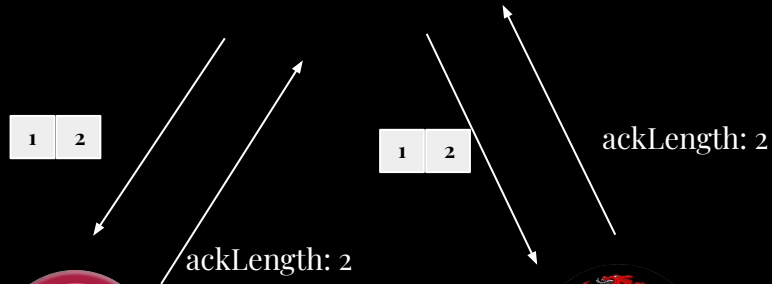


TARGARYEN





STARK



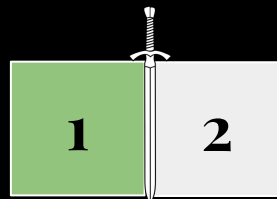
LANNISTER



TARGARYEN



{  : 2,  : 2,  : 2 }



STARK



ackLength: 2

ackLength: 2



LANNISTER



TARGARYEN

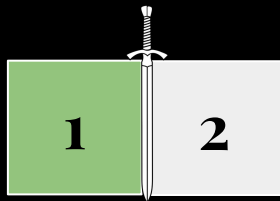


{  : 2,  : 2,  : 2 }

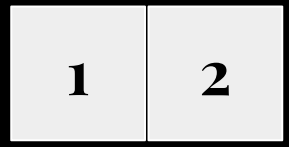
Majorité ?



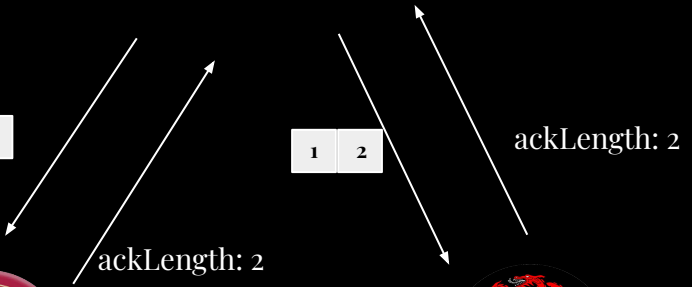
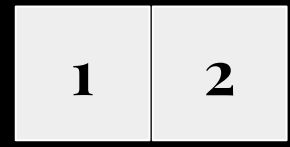
STARK



LANNISTER



TARGARYEN



ackLength: 2

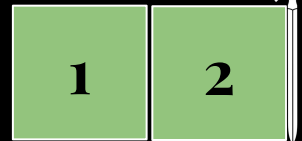
ackLength: 2

{  : 2,  : 2,  : 2 }

Majorité ?



STARK



ackLength: 2

ackLength: 2



LANNISTER



TARGARYEN

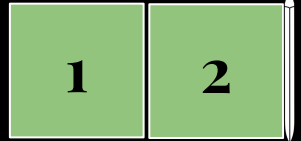


{  : 2,  : 2,  : 2 }

Majorité ?



STARK



commitLength: 2



commitLength: 2



ackLength: 2



LANNISTER



TARGARYEN

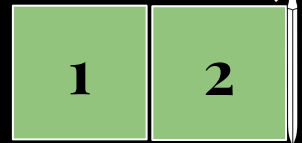


{  : 2,  : 2,  : 2 }

Majorité ?



STARK



commitLength: 2



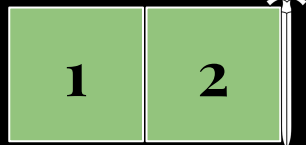
commitLength: 2



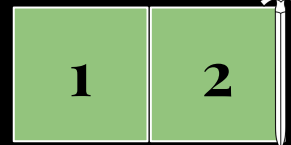
ackLength: 2



LANNISTER



TARGARYEN



GOT CODE ?

```
previousElements = [];  
selectedScopes = [];  
  
scope.$watch(watchExpr, function ngSwitchWatchAction(value) {  
    var i, ii;  
    for (i = 0, ii = previousElements.length; i < ii; ++i) {  
        previousElements[i].remove();  
    }  
    previousElements.length = 0;  
  
    for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
        selectedScopes[i].destroy();  
        previousElements[i] = selected;  
        scope.$eval(selected.$watchExpr, function ngSwitchWatchAction(value) {  
            previousElements.splice(i, 1);  
        });  
    }  
    selectedElements.length = 0;  
    selectedScopes.length = 0;  
  
    if ((selectedTranscludes = ngSwitchController.cases['!' + value] || ngSwitchController.defaultCase)) {  
        scope.$eval(attr.change);  
        forEach(selectedTranscludes, function(selectedTransclude) {  
            var selectedScope = scope.$new();  
            selectedScopes.push(selectedScope);  
            selectedScope.$parent = scope;  
            selectedScope.$watch(selectedTransclude.$watchExpr, function ngSwitchWatchAction(value) {  
                selectedScope.$destroy();  
                selectedScopes.splice(i, 1);  
            });  
            selectedScope.$transclude(selectedTransclude.template, function(clone, scope) {  
                clone.appendTo(selectedScope);  
            });  
        });  
    }  
});
```

```
selectedElements.length = 0;  
selectedScopes.length = 0;
```

```
if ((selectedTranscludes = ngSwitchController.cases['!' + value] || ngSwitchController.defaultCase)) {  
    scope.$eval(attr.change);  
    forEach(selectedTranscludes, function(selectedTransclude) {  
        var selectedScope = scope.$new();  
        selectedScopes.push(selectedScope);  
        selectedScope.$parent = scope;  
        selectedScope.$watch(selectedTransclude.$watchExpr, function ngSwitchWatchAction(value) {  
            selectedScope.$destroy();  
            selectedScopes.splice(i, 1);  
        });  
        selectedScope.$transclude(selectedTransclude.template, function(clone, scope) {  
            clone.appendTo(selectedScope);  
        });  
    });  
}
```

Ce qui nous manque pour finir **RAFT**



Ce qui nous manque pour finir **RAFT**

Le merge de log

- Le leader note où les followers en sont
- Le leader n'envoie que la section manquante du log aux follower
- Le leader traque le term de chaque entry dans le log
- Les followers truncate les logs non commités pendant un crash



MERC