

Ecosistema Hadoop

Máster en Data Science y Big Data

Jorge López-Malla Matute

jlmalla@geoblink.com

Febrero 2022

Presentación

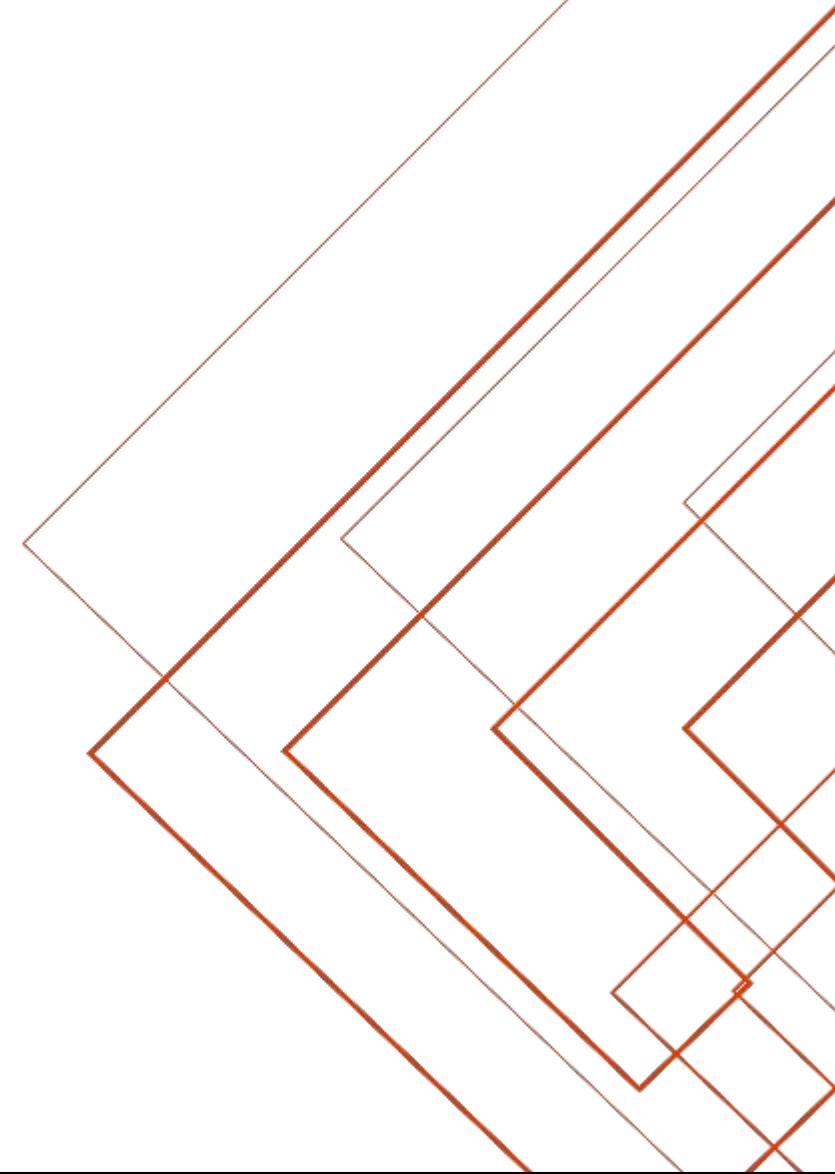
- Jorge López-Malla Matute
- Puesto actual:
 - Senior Data Engineer en Geoblink
- Experiencia docente:
 - Profesor en diversos Masters de Big Data durante los últimos 6 años
 - Profesor de Tecnologías Masivas en ICAI
- Años trabajados en Big Data: 9 años



Índice

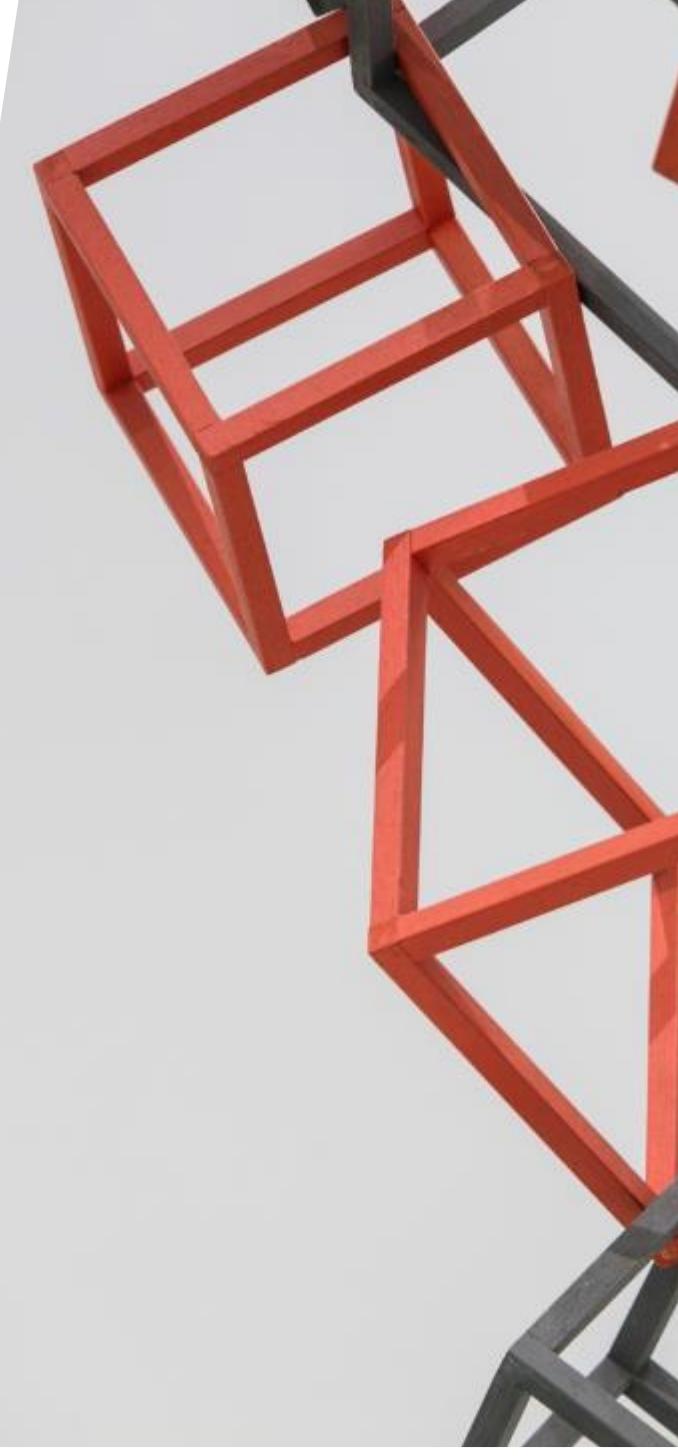
Contenido

1. Introducción
 - Repaso de Conceptos
 - Historia de Hadoop
2. Hadoop Core
 - Apache HDFS
 - Apache Map & Reduce
 - Apache YARN
3. Ecosistema Hadoop
 - Apache Hive
 - Apache HBase



Introducción

Repaso de conceptos



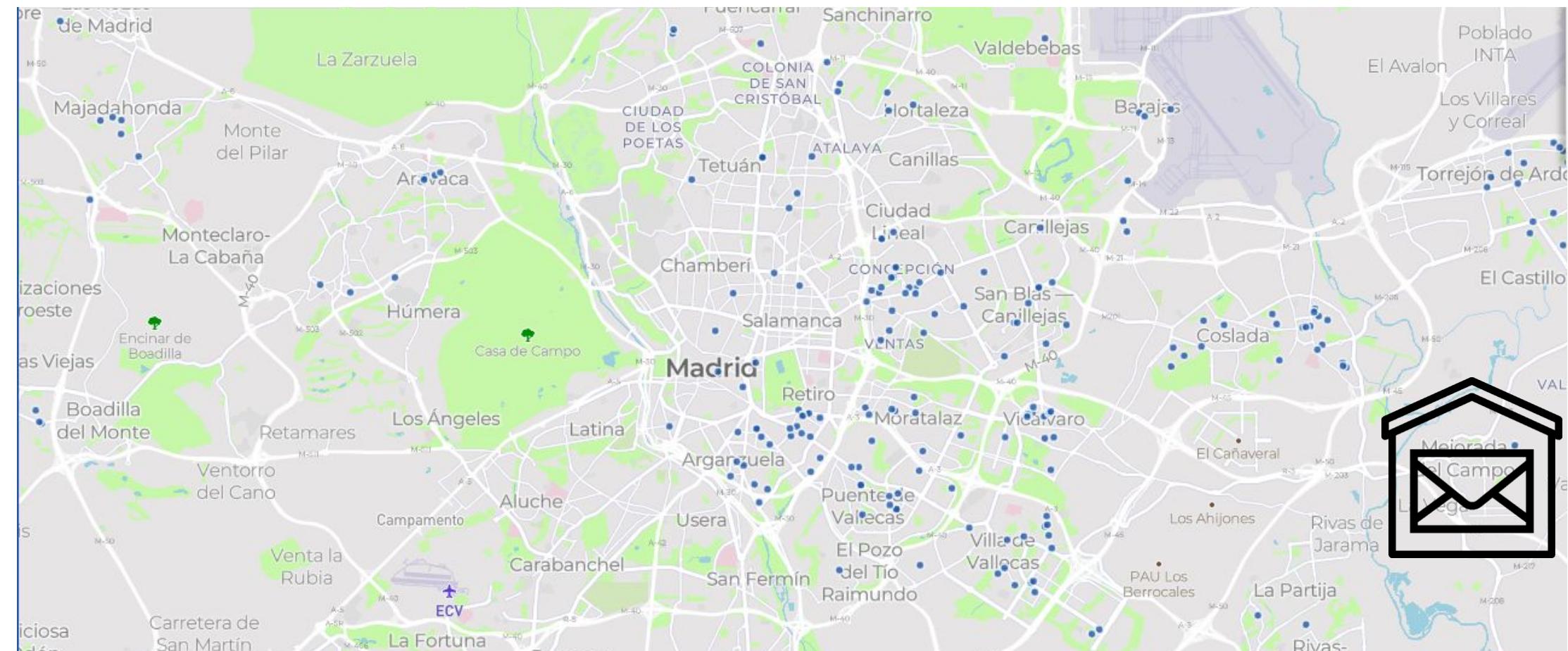
Tipos de escalabilidad

- Cuando la carga de un sistema informático, ya sea de cómputo o de cantidad de datos, supera las características del mismo este tiene que **escalar**.
- Si no se toman medidas de diseño o tecnológicas previas **escalar un sistema informático es costoso en tiempo y en dinero**
- En un mundo tan digitalizado como el actual todo sistema informático tiene que estar preparado para escalar sin previo aviso
- Los sistemas informáticos pueden escalar de dos formas **verticalmente** y **horizontalmente**
- Las soluciones **Big Data**, por norma, **tienen que tender a una escalabilidad horizontal**

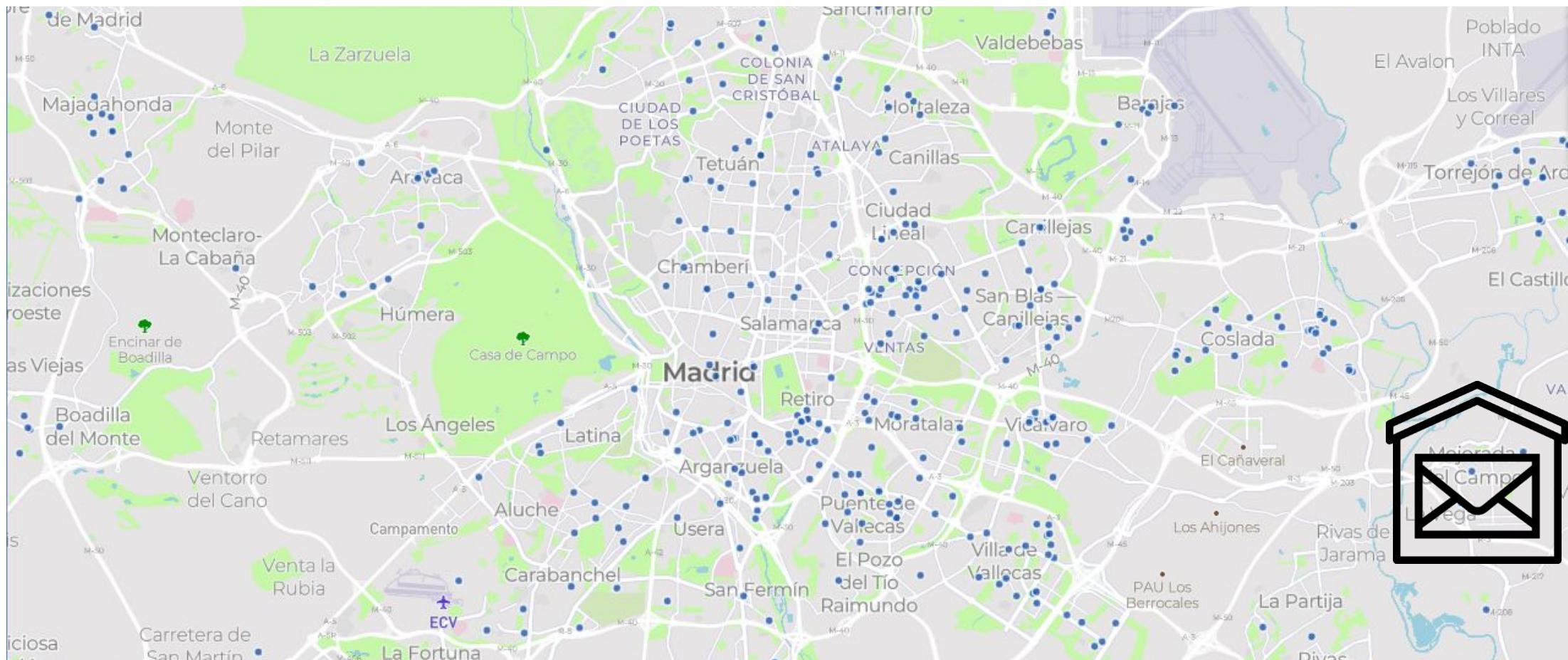
Escalabilidad vertical

- Un sistema escala verticalmente cuando su estrategia para el escalado se basa en la cantidad de recursos de una máquina no en la cantidad de máquinas
- En este tipo de estrategias se copia el mismo sistema en una máquina con más recursos con lo que soporta más carga
- Durante esa copia se produce una parada del servicio
- Las máquinas más grandes, normalmente, suelen costar más dinero que la suma de recursos en máquinas más pequeñas
- Siguen siendo sistemas con un único punto de fallo
- Como punto positivo simplifican la ampliación del sistema informático

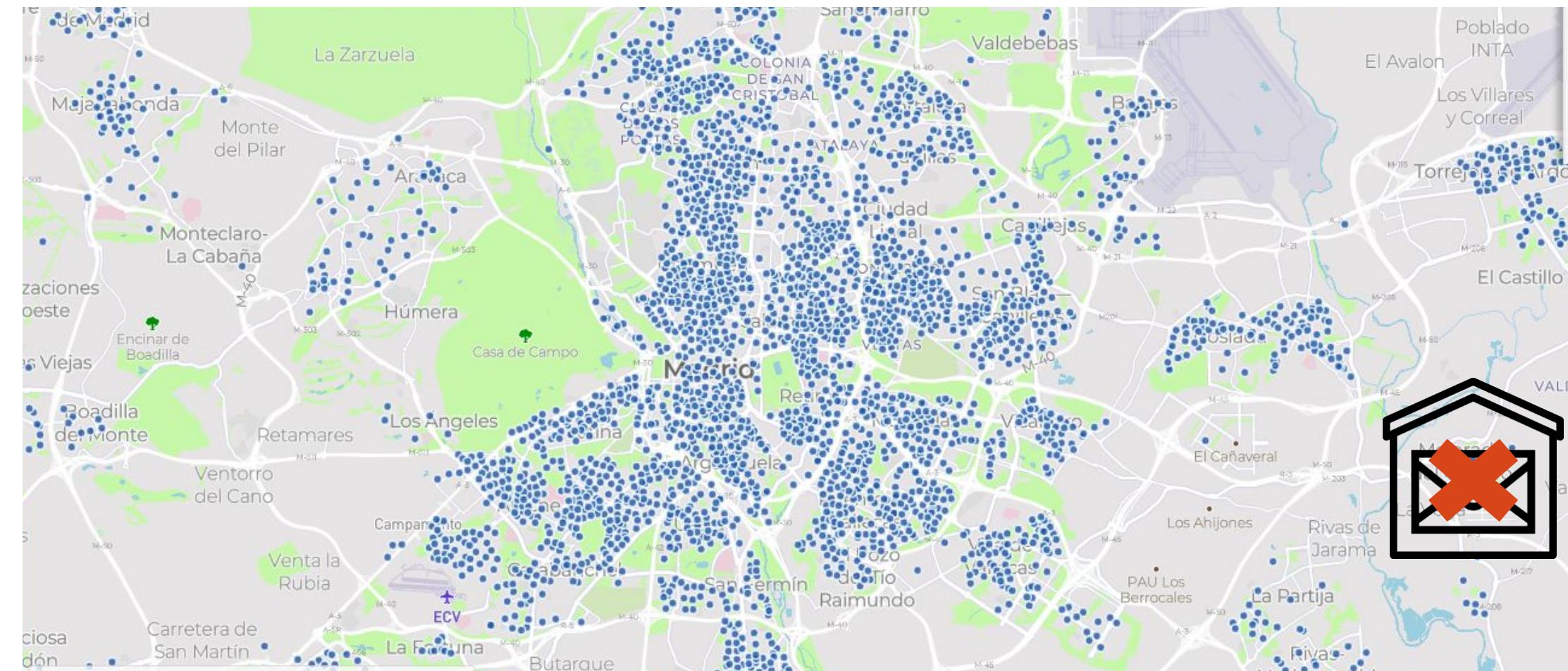
Escalabilidad vertical



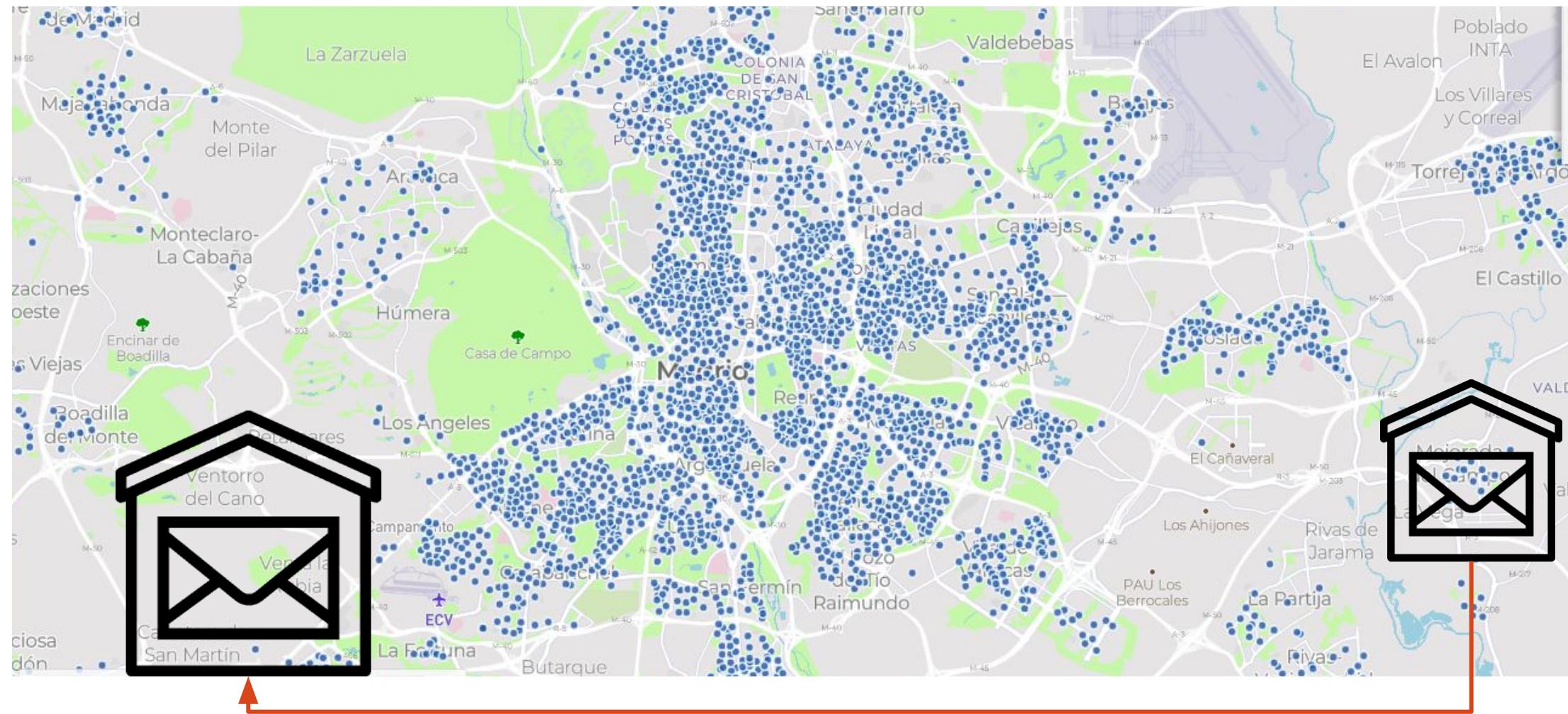
Escalabilidad vertical



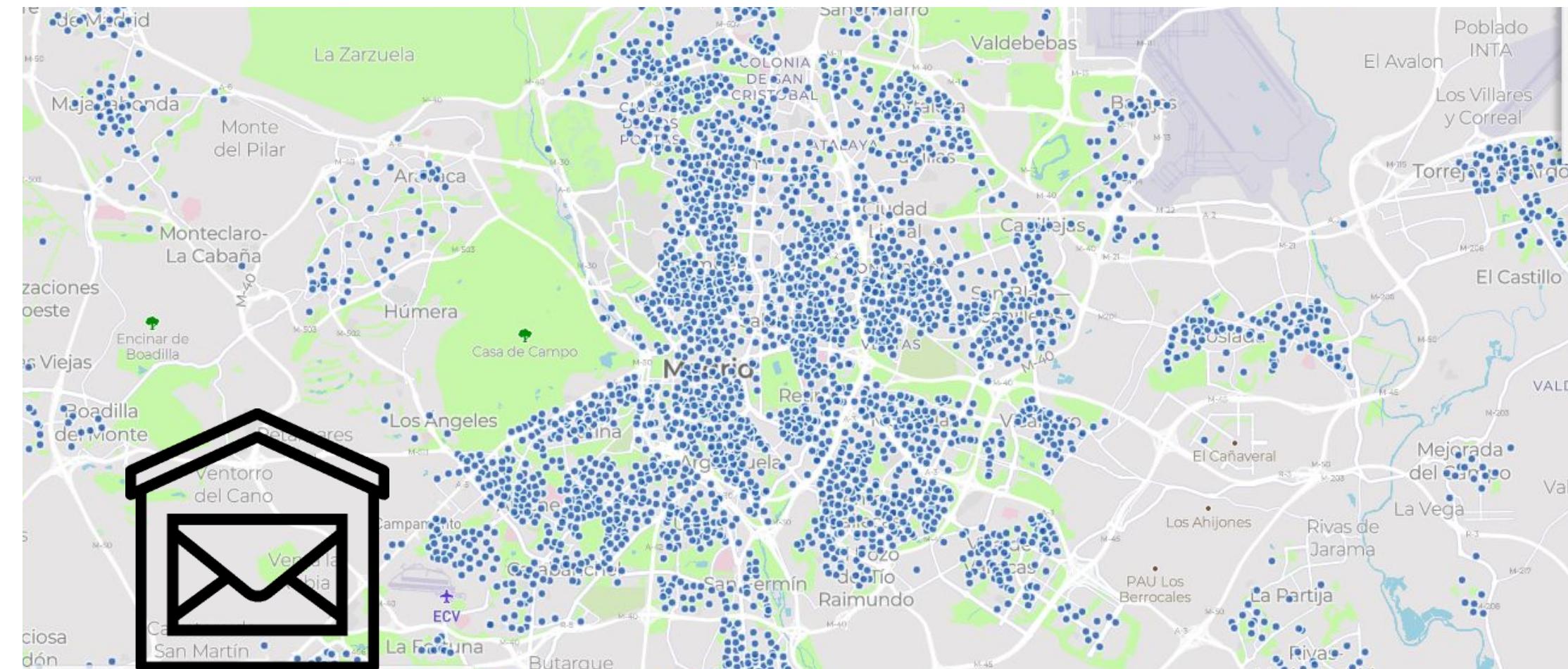
Escalabilidad vertical



Escalabilidad vertical



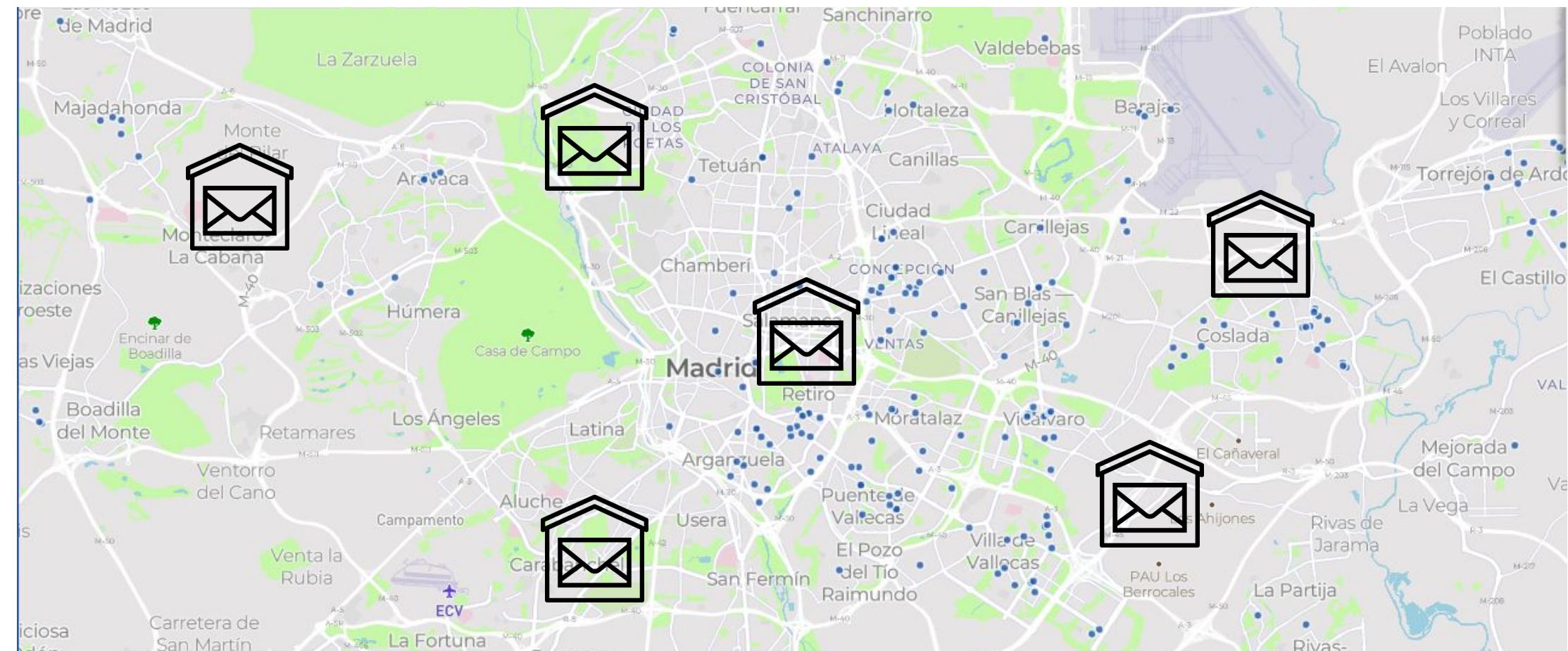
Escalabilidad vertical



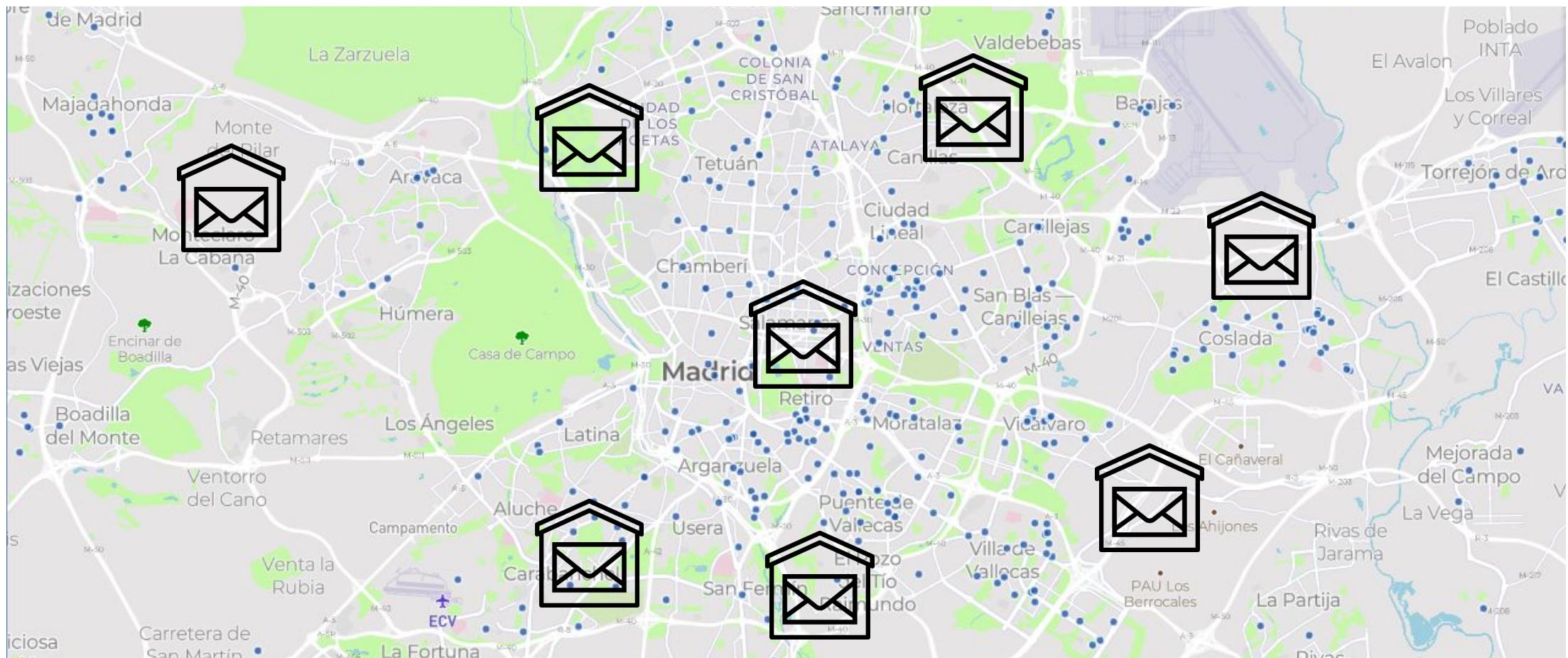
Escalabilidad horizontal

- Un sistema escala horizontalmente cuando su estrategia para el escalado se basa en la cantidad de máquinas en vez de los recursos de cada máquina
- En este tipo de estrategias la carga se distribuye entre distintas máquinas por lo que ampliarlo “sólamente” requiere añadir una máquina
- Durante ese escalado no se produce una parada del servicio
- Se pueden elaborar estrategias para evitar los puntos únicos de fallo
- En contraposición al escalado vertical añaden nuevos puntos de ataques y complican la tecnología de los sistemas informáticos

Escalabilidad horizontal



Escalabilidad horizontal



Escalabilidad horizontal



Historia de Hadoop



Historia de Hadoop

- **Apache Hadoop** fue el primer ecosistema Open Source Big Data.
- Empieza su desarrollo por Yahoo! en 2005
- Fue el primer ecosistema Open Source en el que las soluciones de almacenamiento, procesamiento y gestión del mismo estaban en el mismo core
- **HDFS** para el almacenamiento, **Map&Reduce** para el procesamiento y JobTracker para la gestión del procesamiento
- Nació como una solución para **grandes volúmenes** de datos **sin estructura fija** y **procesados “rápidamente”** (para la época) resolviendo problemas en **Batch**

Historia de Hadoop

- En 2005 empezaba a haber un **problema de almacenamiento** y de gestión del mismo.
- Los discos magnéticos son baratos y el cuello de botella del procesamiento está en las comunicaciones
- **Escalar verticalmente** los sistemas empieza a no ser una solución
- La primera solución de almacenamiento concurrente Open Source que permite escalabilidad horizontal, **HDFS**, se convierte en la piedra angular del ecosistema
- Las siguientes soluciones se basan en el principio de que el almacenamiento es que **el disco es "gratis"**

Historia de Hadoop

- Cuando se ve el potencial de la analítica de datos se crean proyectos alrededor del proyecto Hadoop
- Se empiezan a tratar otras problemas como las bases de datos NOSQL y el Machine Learning aplicado al Big Data
- Hadoop empieza como una herramienta pensada únicamente para desarrolladores
- La alta curva de aprendizaje hace que emprendan proyectos para simplificar el desarrollo del procesamiento

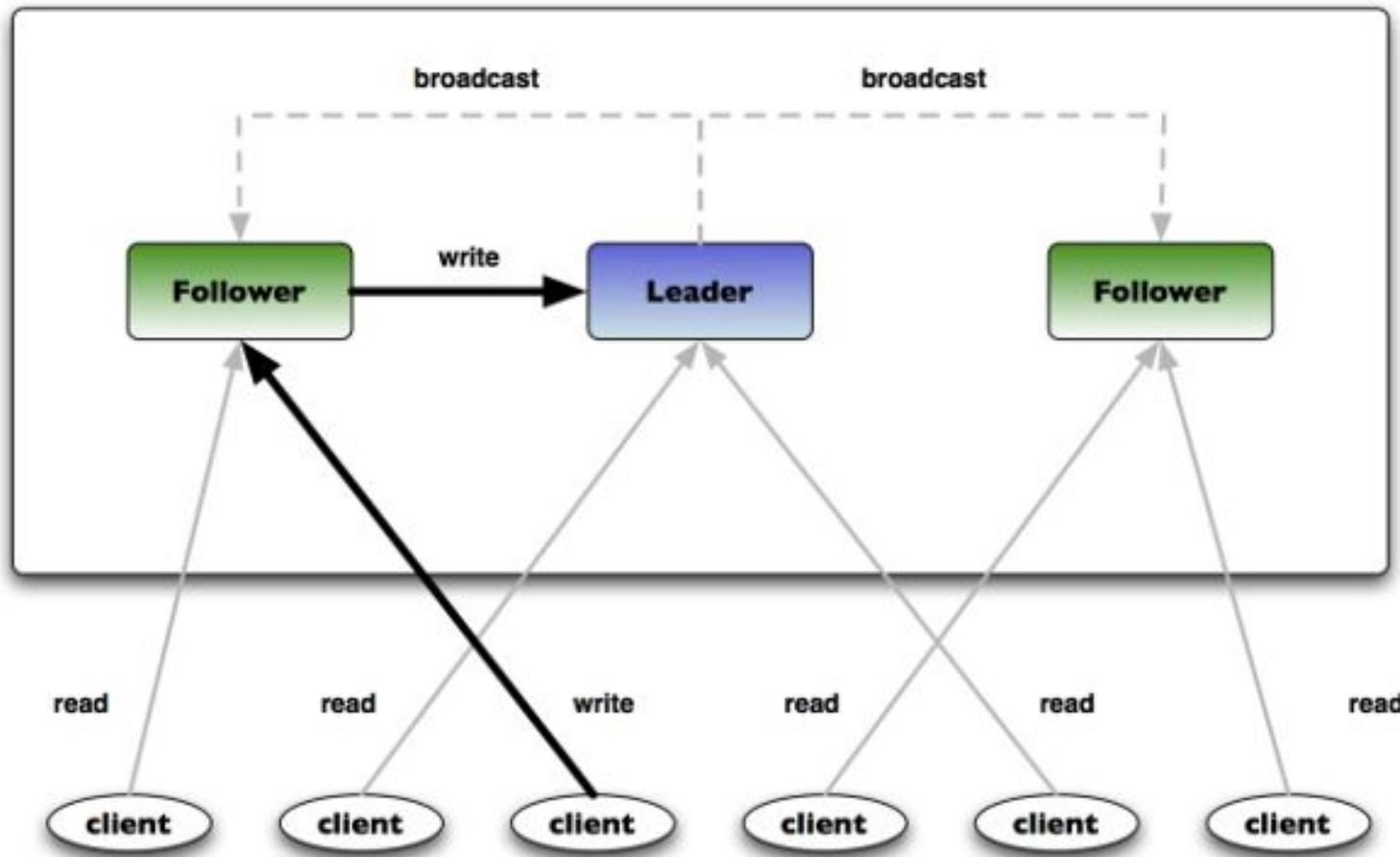
Historia de Hadoop

- El ecosistema se asienta en tres proyectos en los que se basan sus distintas soluciones:
 - **HDFS**: Sistema de ficheros de Hadoop. Proyectos de almacenamiento como HBase se basan en él.
 - **MapReduce**: Modelo de programación distribuida. Los distintos proyectos de computación del ecosistema resuelven casuísticas usando este modelo.
 - **YARN**: Gestor de aplicación de Hadoop. Es el encargado de gestionar las distintas tareas de MapReduce. También es capaz de gestionar otros proyectos de cómputo masivo como por ejemplo Spark

Historia de Hadoop

- Al tratar con cantidades masivas de datos todos los servicios tienen que estar preparados para el fallo
- La mayoría de tecnologías del ecosistema están preparadas para la tolerancia a fallos mediante la alta disponibilidad
- Las soluciones tecnológicas se basan en un componente core del ecosistema Apache Zookeeper
- Apache Zookeeper es un bus distribuido donde los distintos nodos pueden escribir de una forma coordinada y atómica

Historia de Hadoop



Hadoop Core

Apache HDFS



HDFS - Definición

- HDFS (Hadoop Distributed File System) es el sistema de ficheros distribuido de Hadoop
- Fue la primera solución de almacenamiento del ecosistema Hadoop, se basa en el sistema de ficheros de Google
- Como la mayoría de componentes del ecosistema de Hadoop HDFS tiene una arquitectura maestro esclavo.
- Las propiedades de almacenamiento se configuran a nivel de fichero pudiendo tener distintos ficheros dentro de la misma carpeta con configuraciones distintas.
- Soporta varios tipos de ficheros

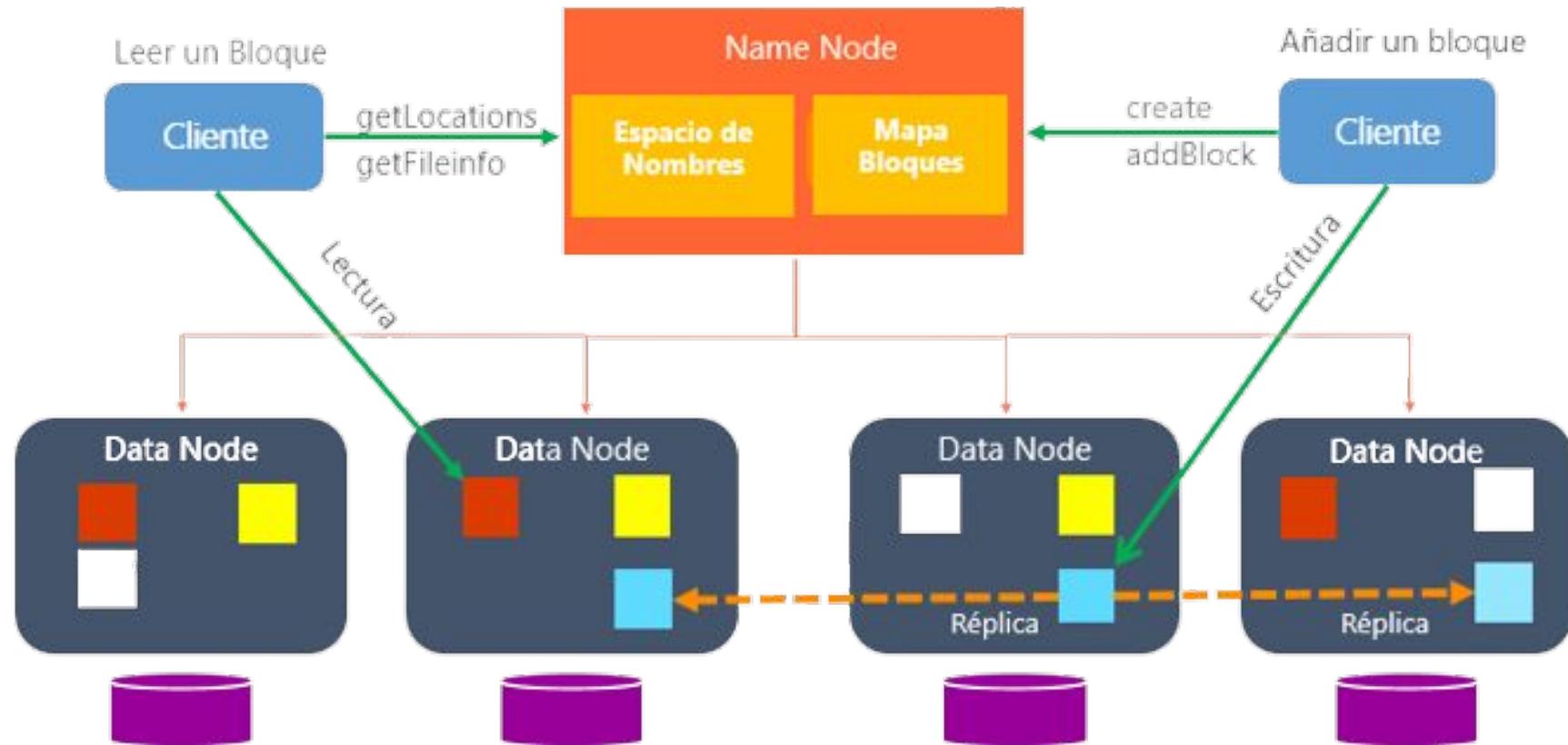
HDFS - Definición

- **Namenode:** Proceso maestro, no almacena la información si no la metainformación de cada uno de los distintos ficheros de HDFS.
 - Aunque no guarde ninguna información todas las operaciones del cluster de HDFS pasan por él
 - No requiere de muchos recursos de espacio
- **Datanodes:** Procesos esclavos, suele haber más de un datanode por cluster. Almacenan realmente los fragmentos de información que les corresponden.
 - Tras preguntar al **Namenode** todos los procesos de lectura y escritura preguntan/envían información a los **Datanodes**
 - Requiere de muchos recursos de espacio
- **Secondary Namenode:** Proceso que se encarga de hacer snapshot temporales, pero no es un namenode de backup

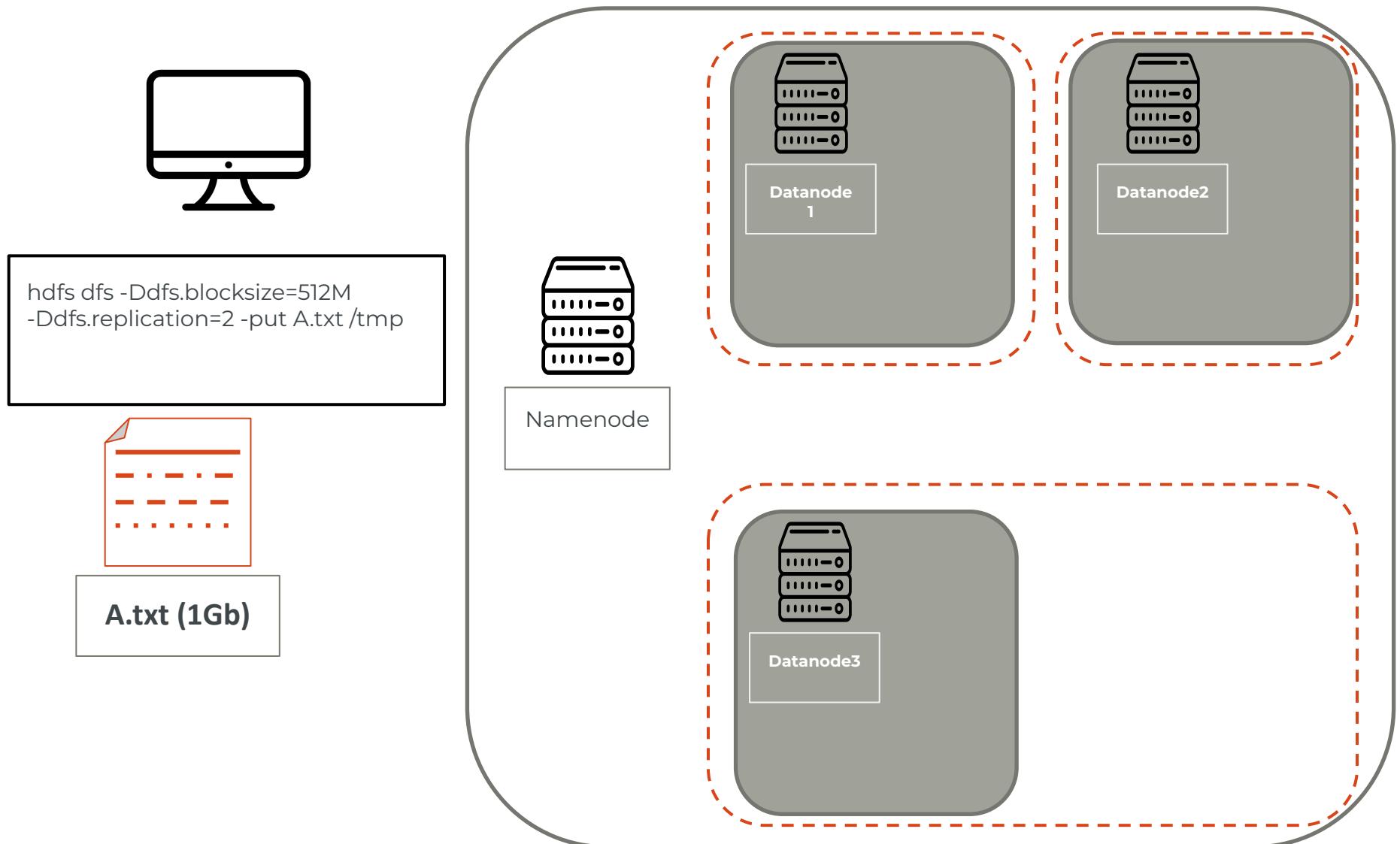
HDFS - Definición

- Para HDFS un fichero se descompone en diversos bloques en los que se fija un tamaño de bloque máximo
- Si el tamaño del fichero no es divisible por el tamaño del bloque el último bloque tiene un tamaño menor
- *El disco es gratis*
- Los problemas de lecturas concurrentes y de fallo en comunicaciones se resuelve mediante la réplica de bloques

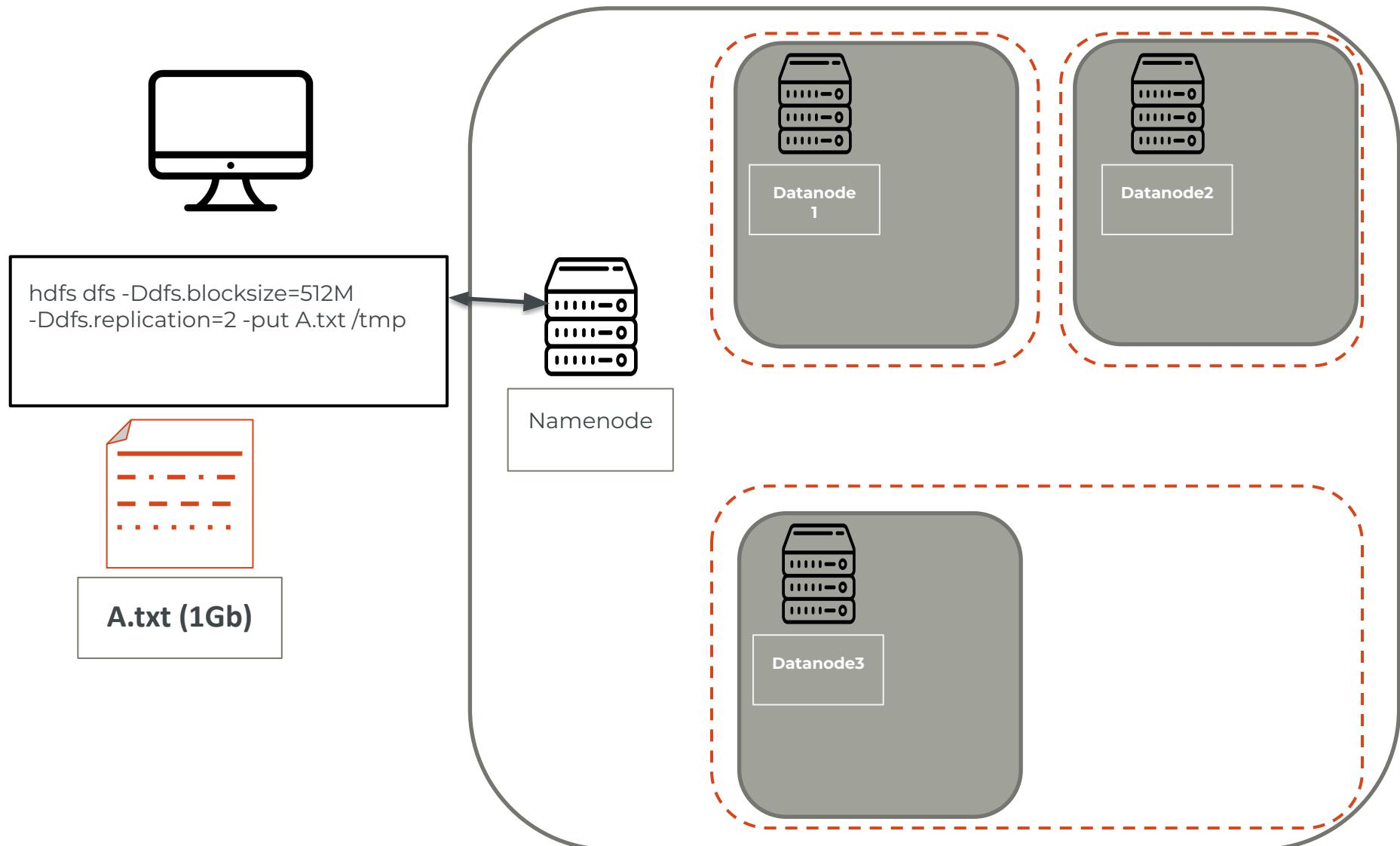
HDFS - Funcionamiento



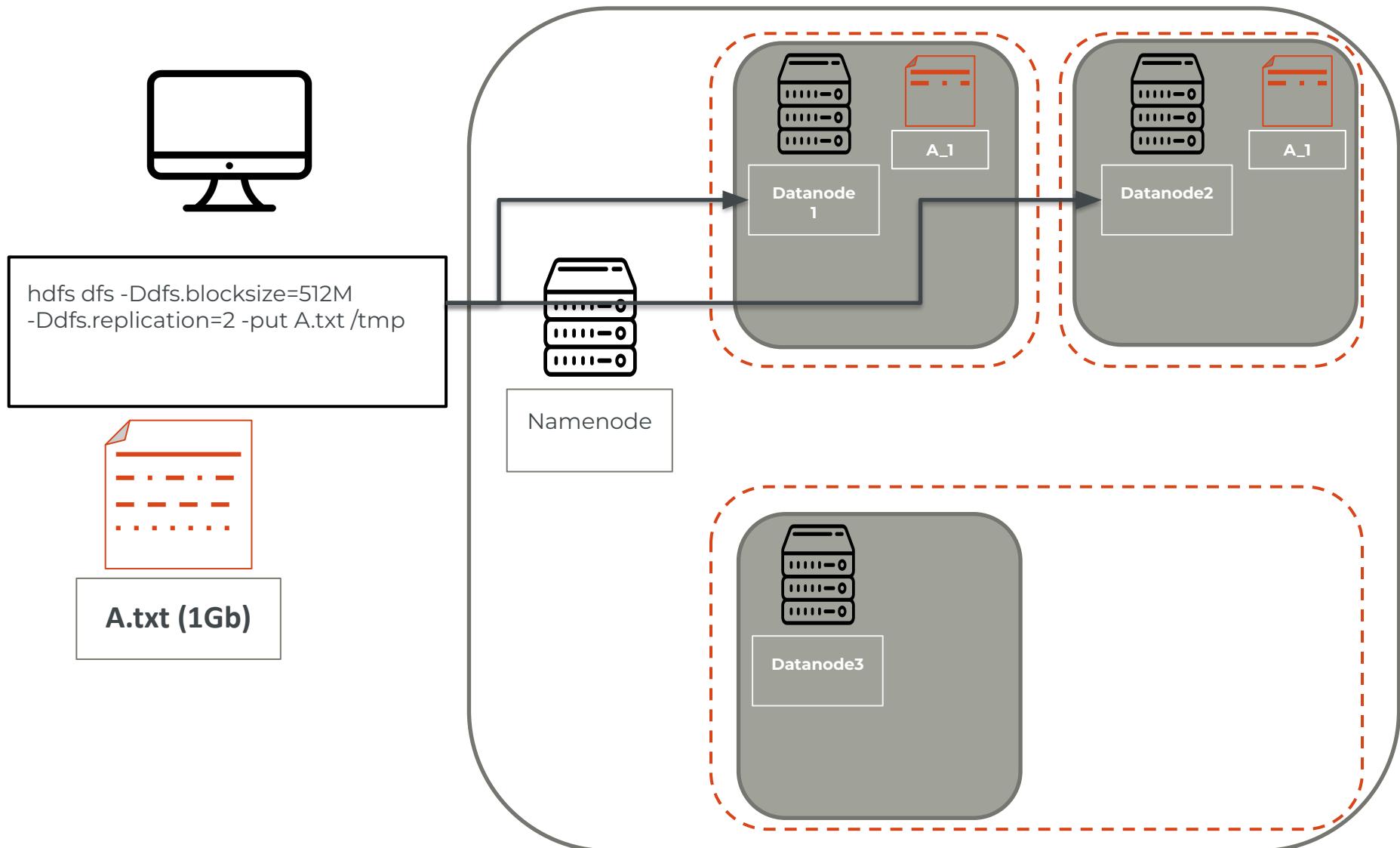
HDFS - Funcionamiento



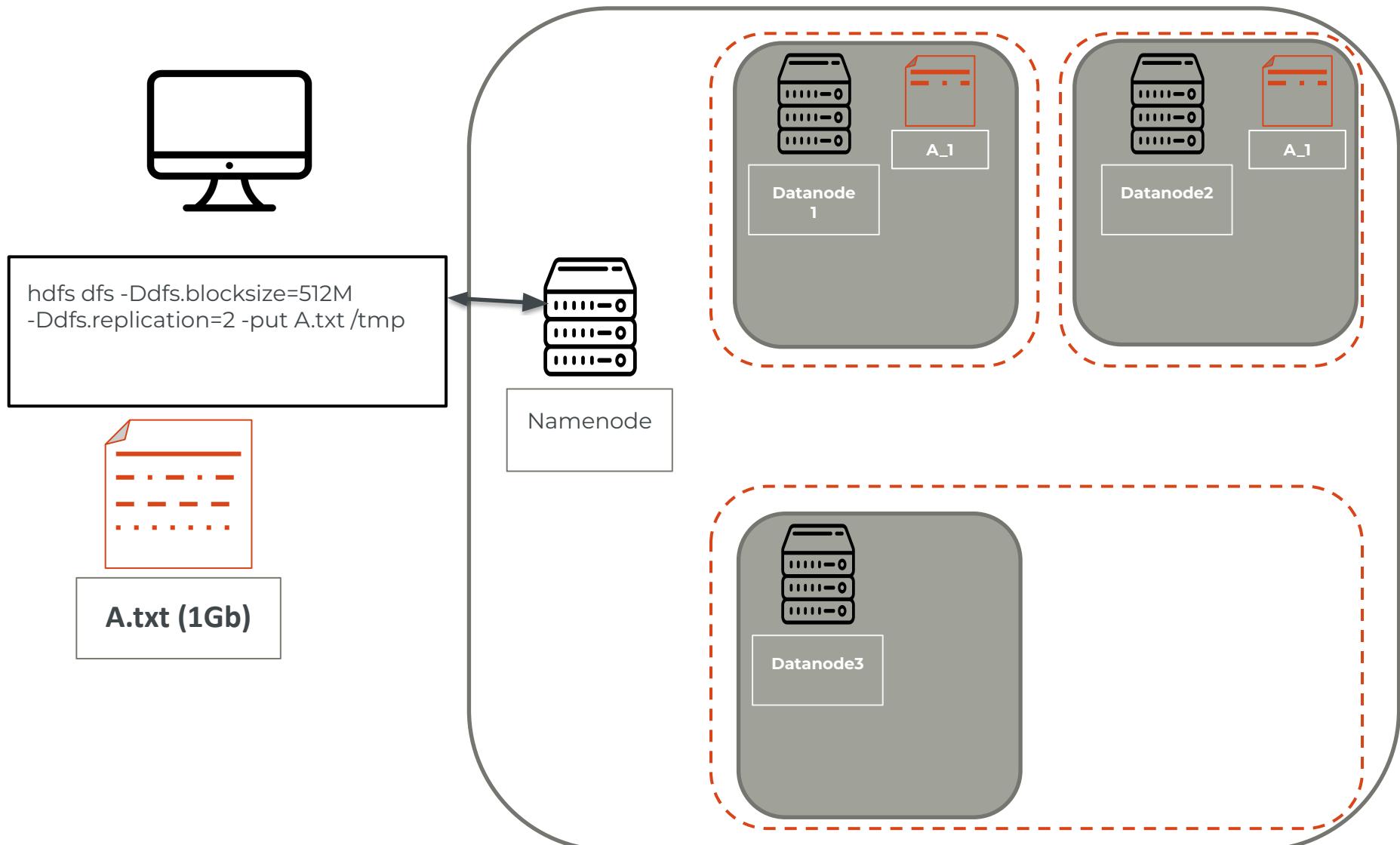
HDFS - Funcionamiento



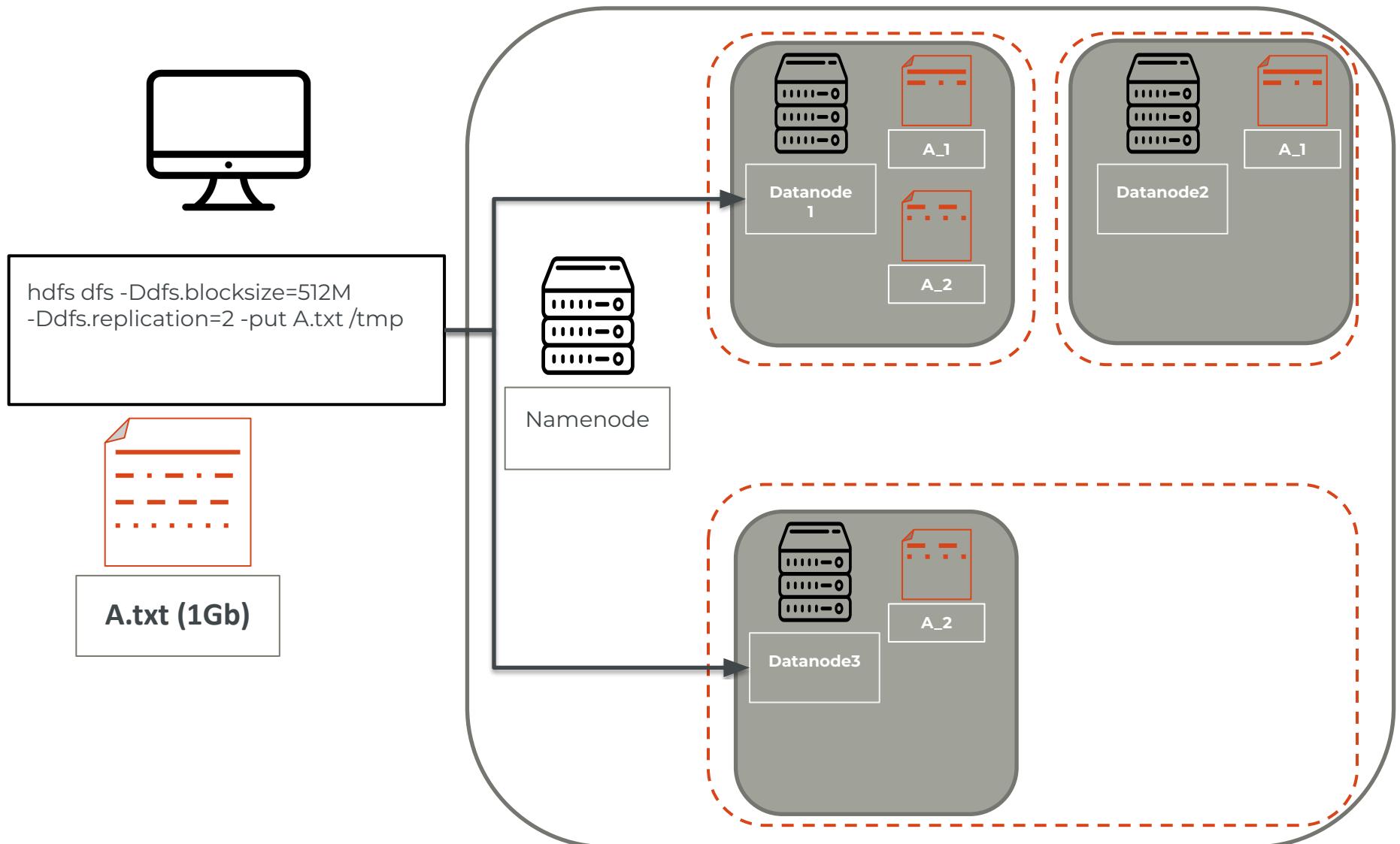
HDFS - Funcionamiento



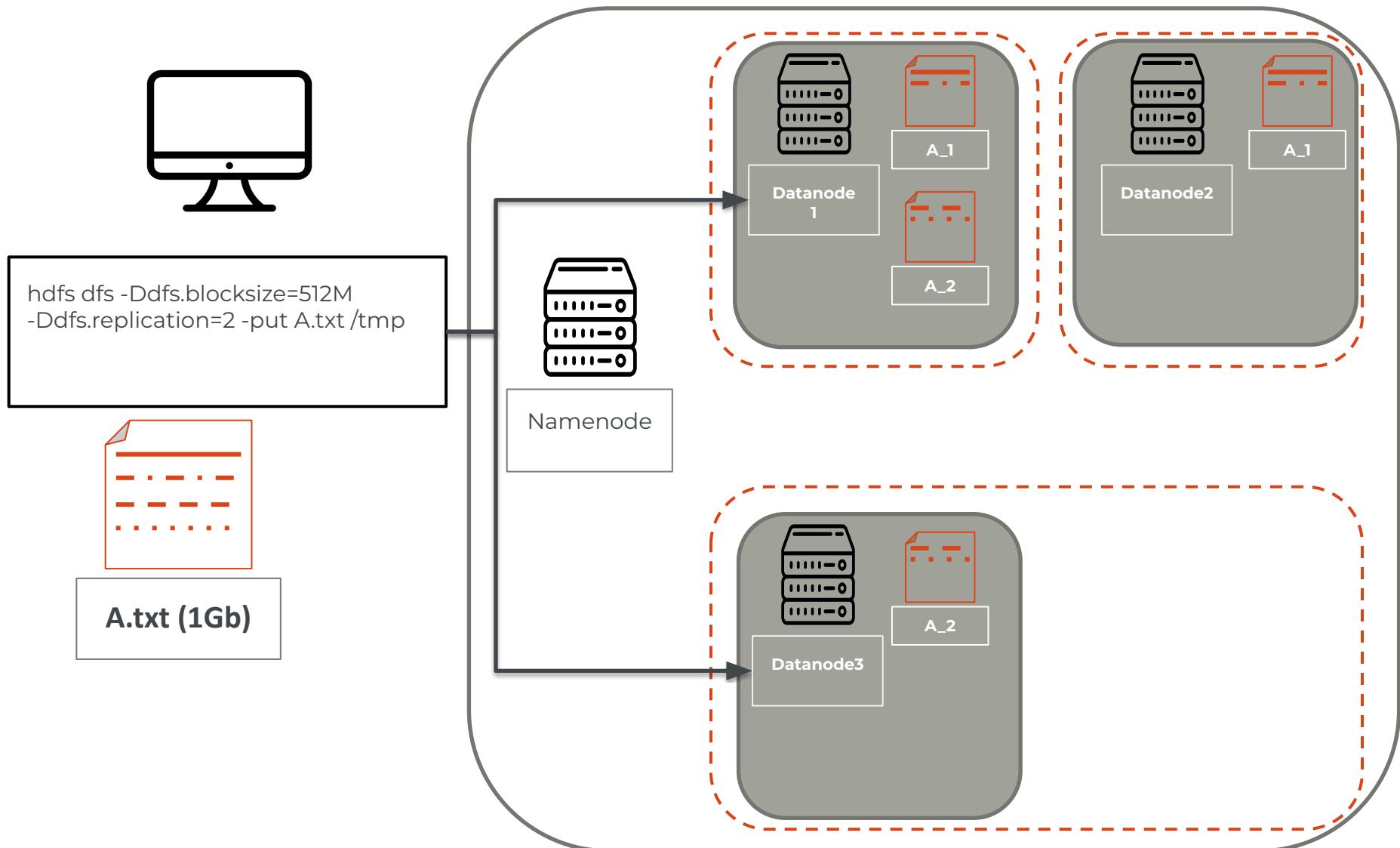
HDFS - Funcionamiento



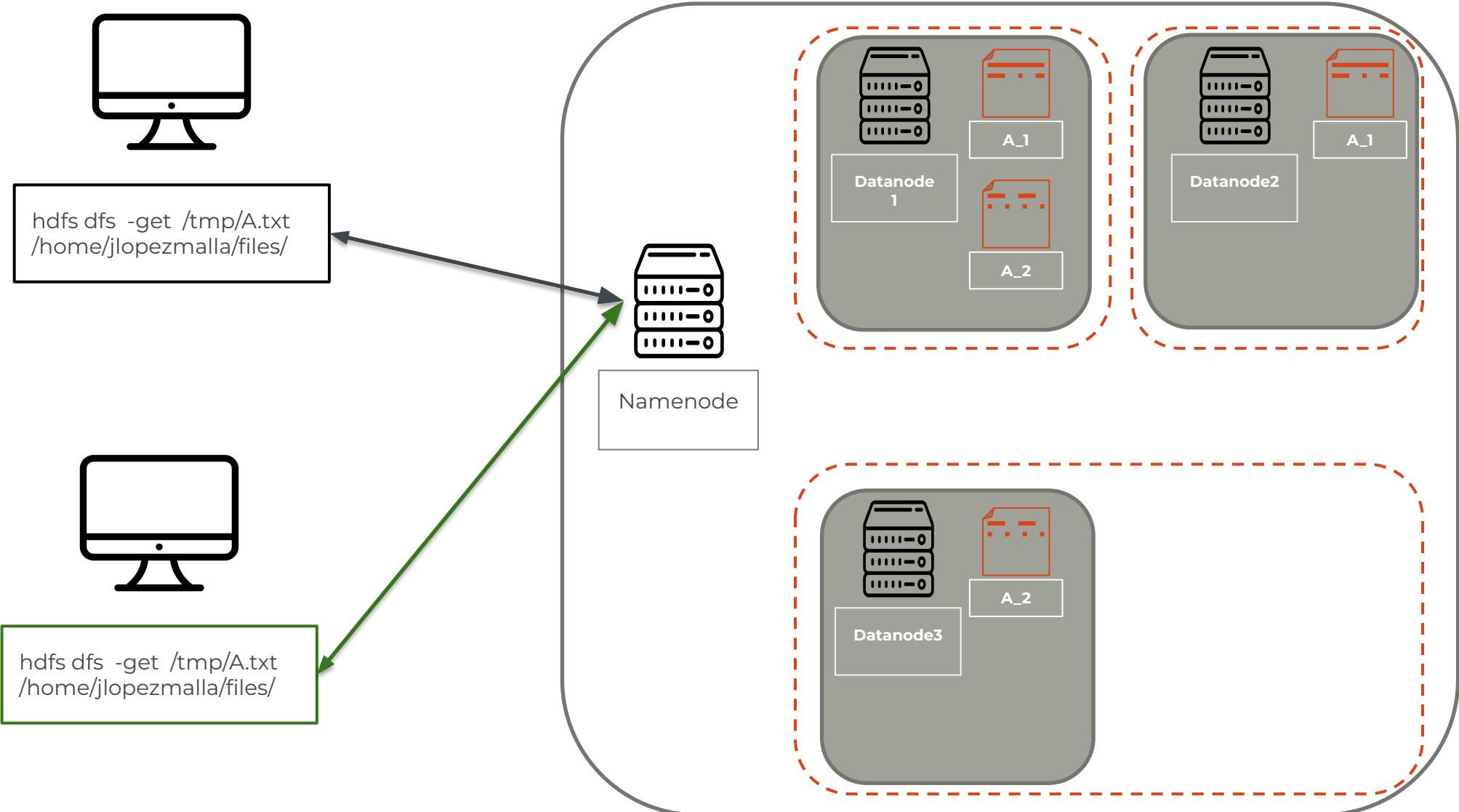
HDFS - Funcionamiento



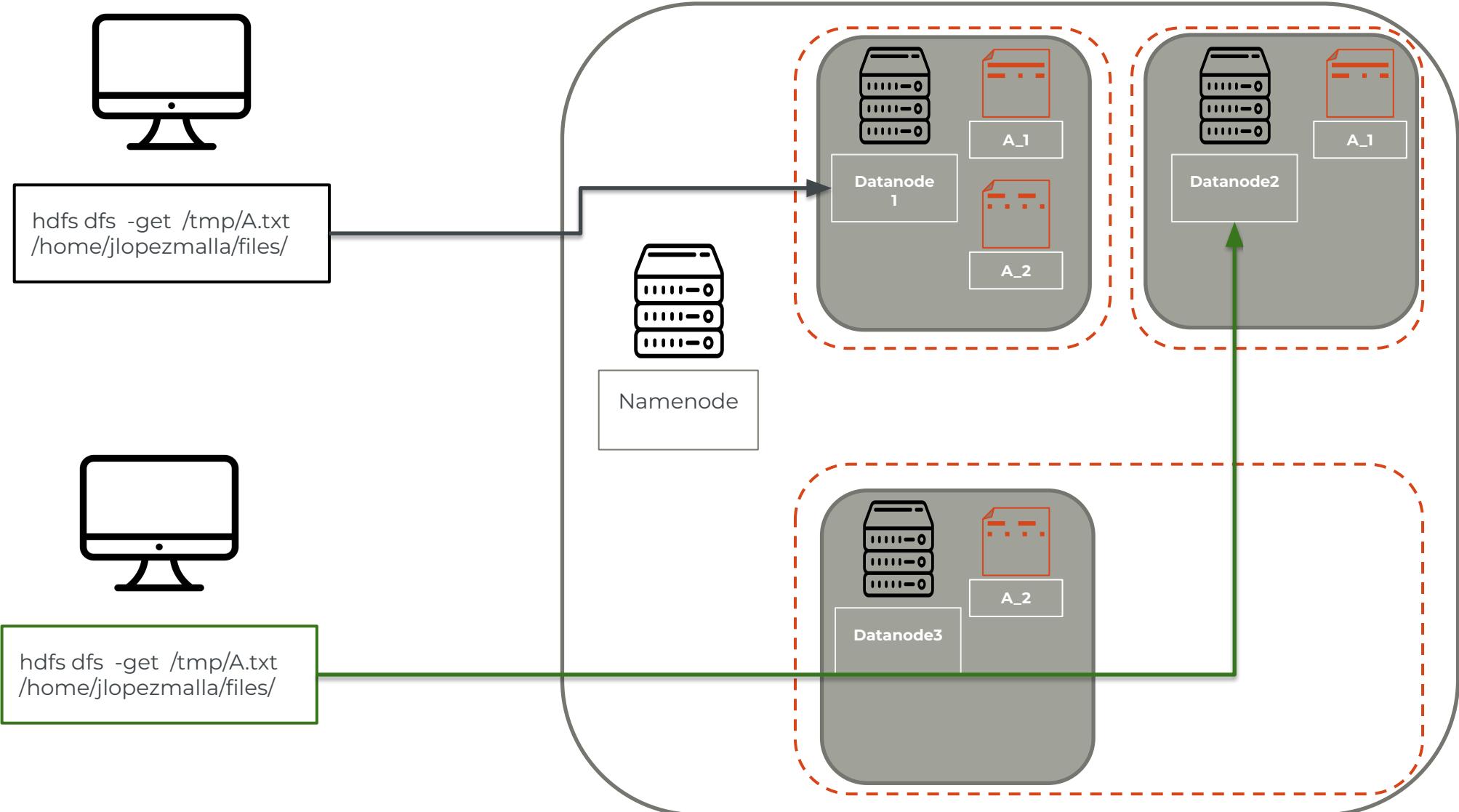
HDFS - Funcionamiento



HDFS - Funcionamiento

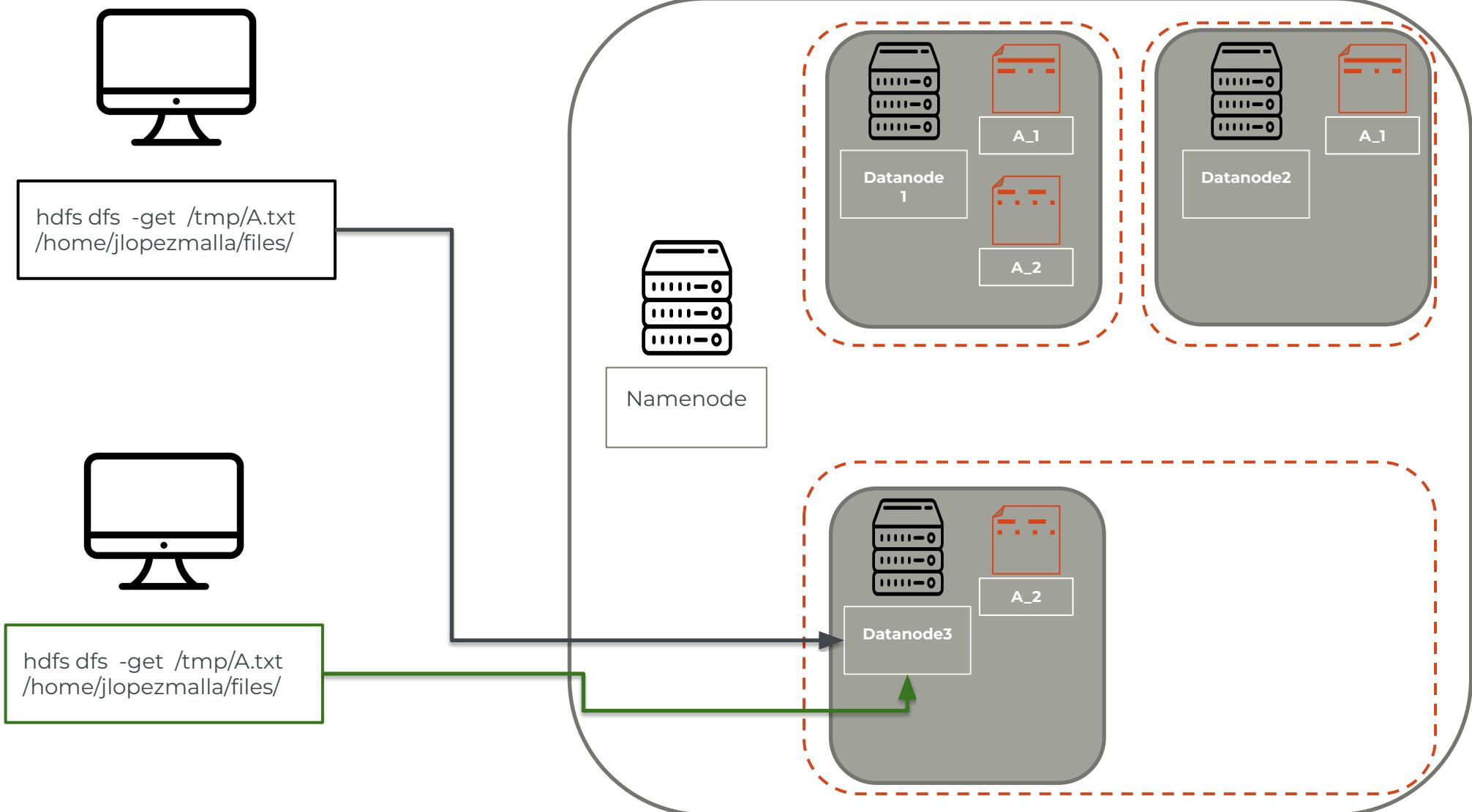


HDFS - Funcionamiento

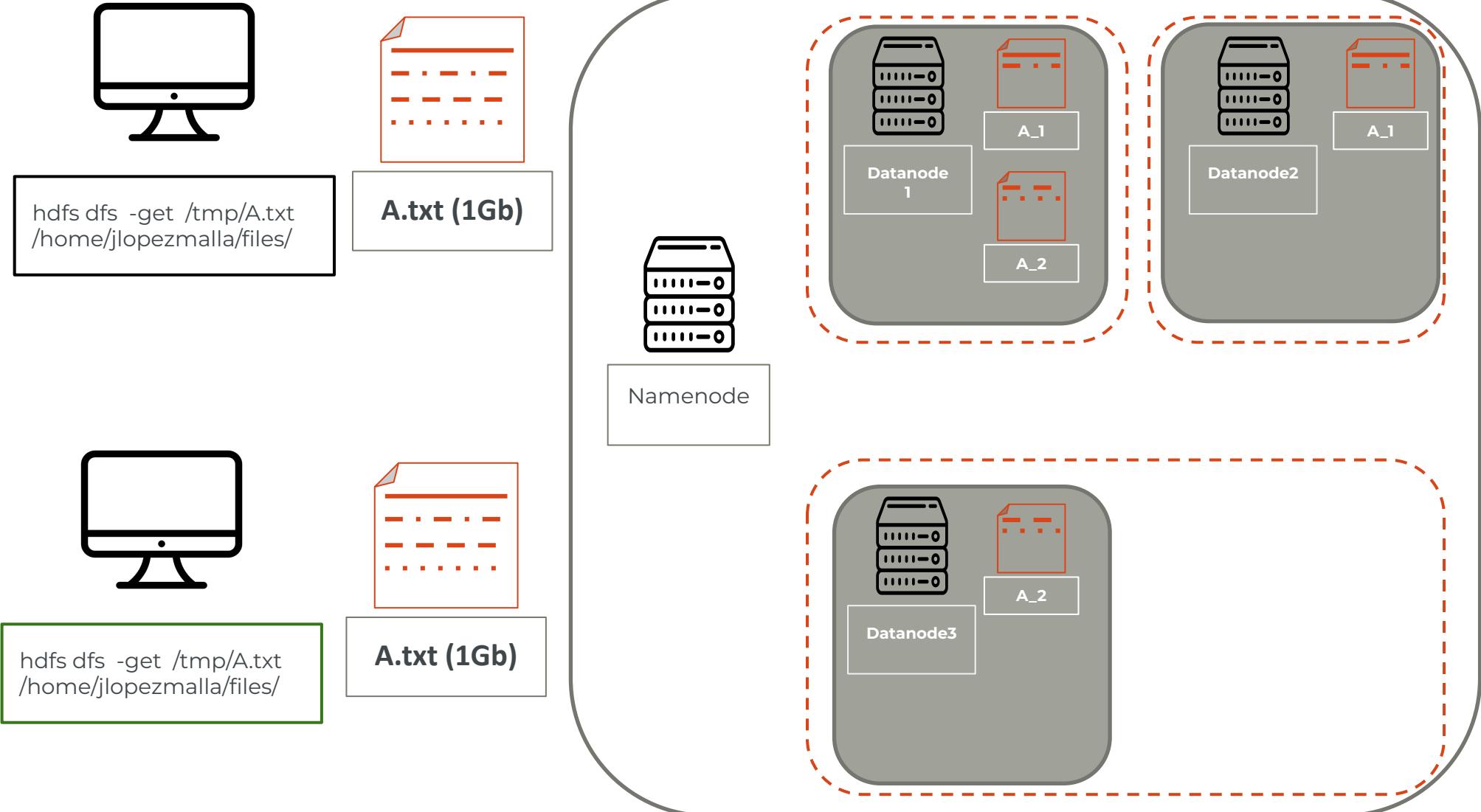


Hadoop

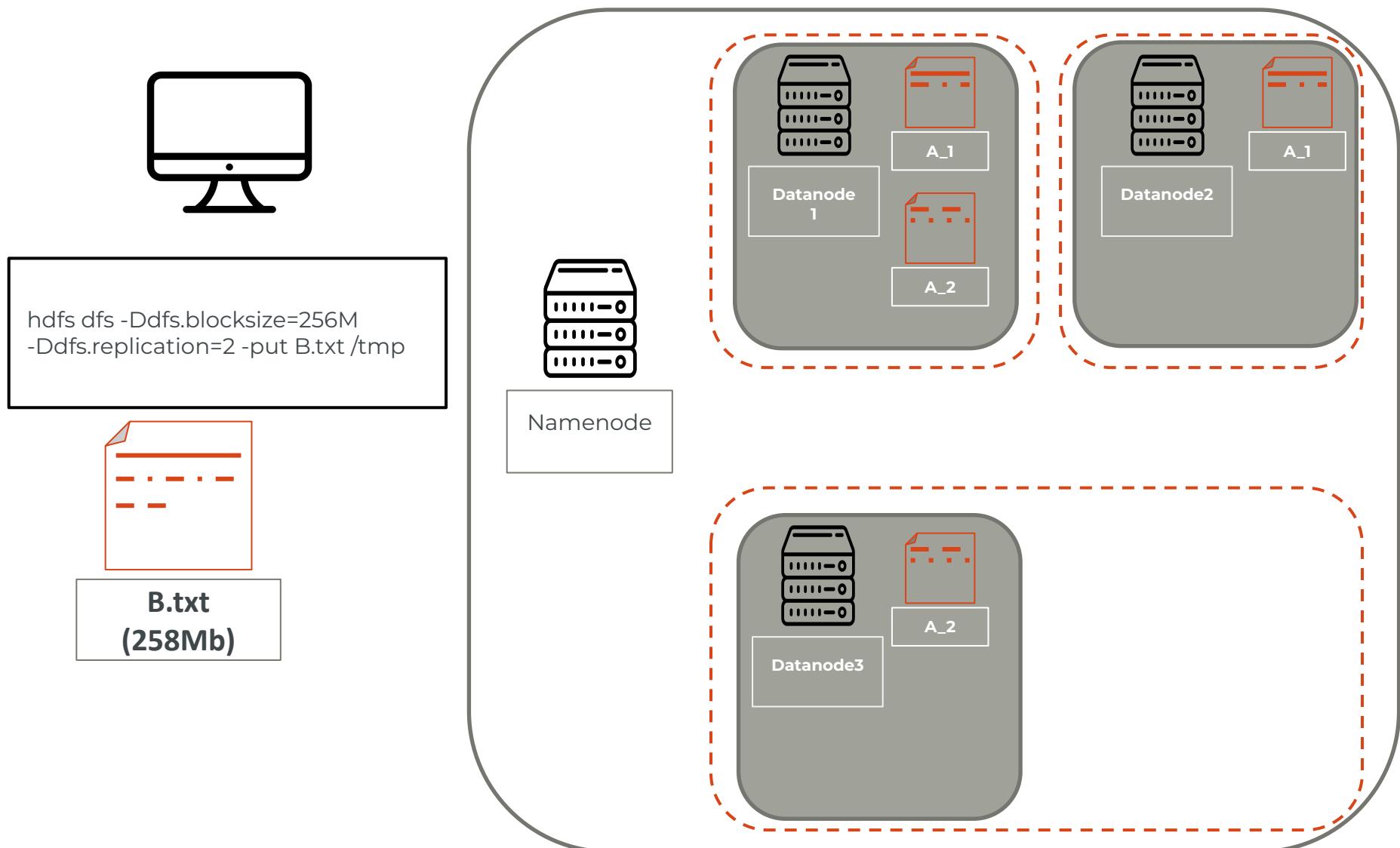
HDFS - Funcionamiento



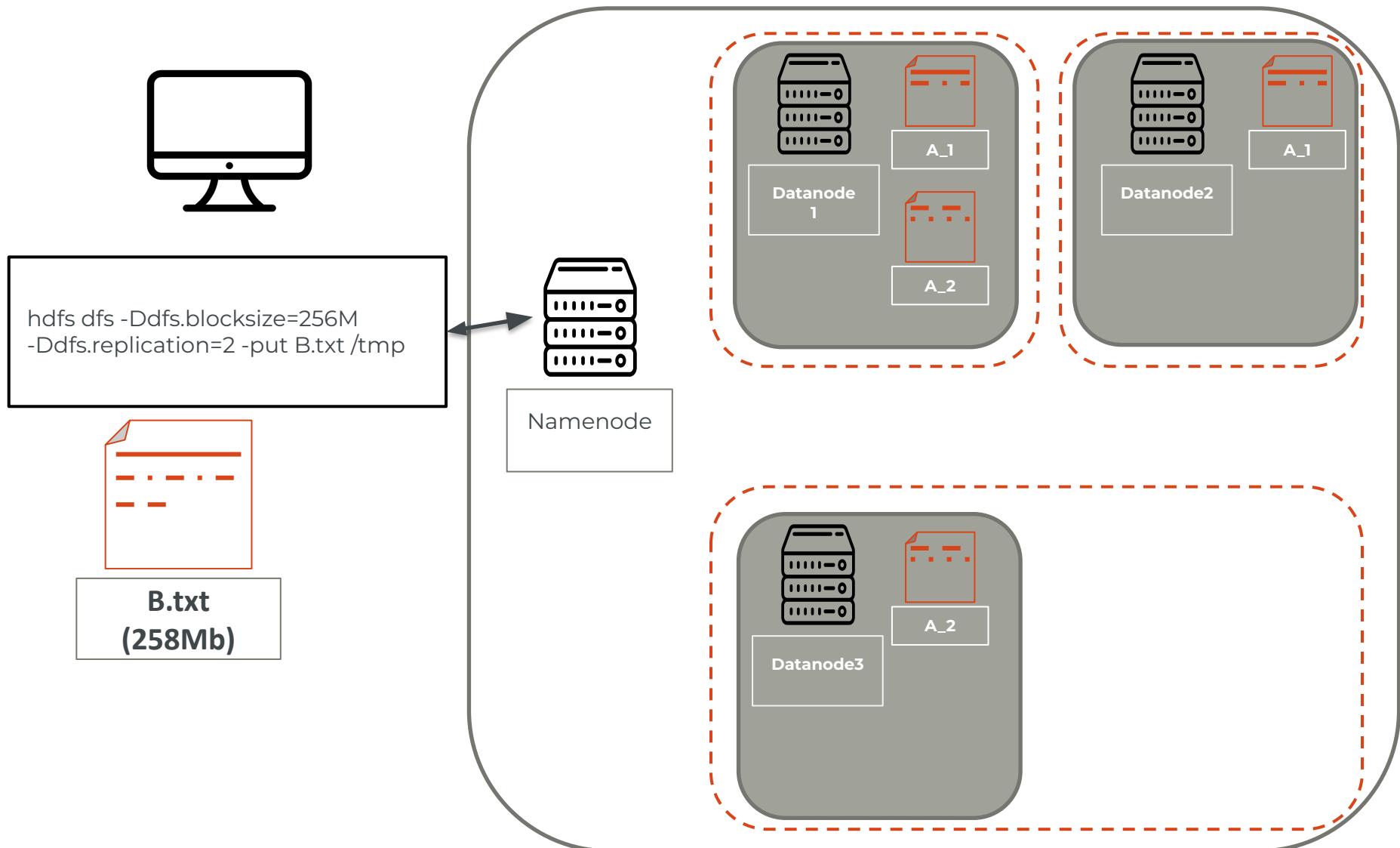
HDFS - Funcionamiento



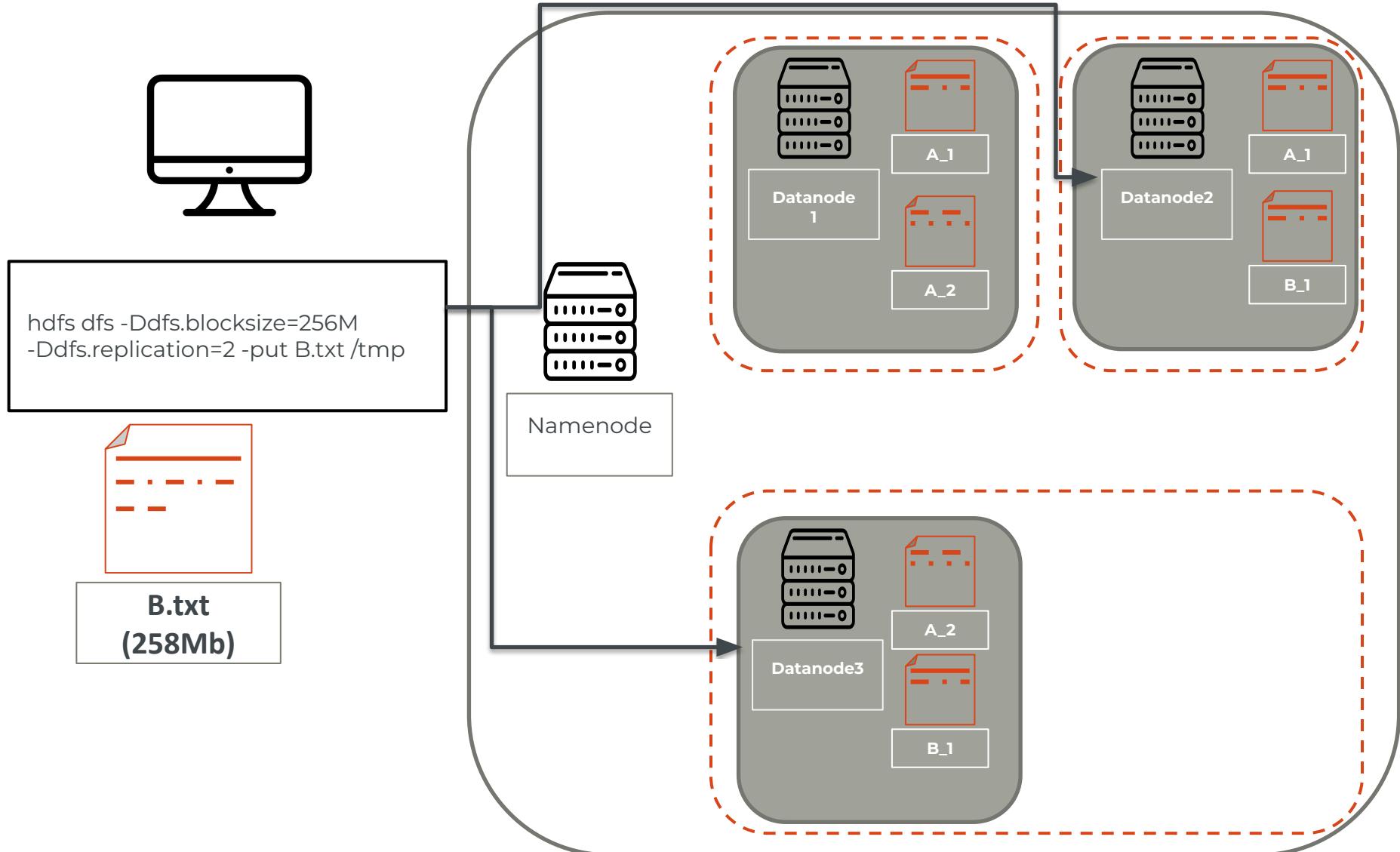
HDFS - Funcionamiento



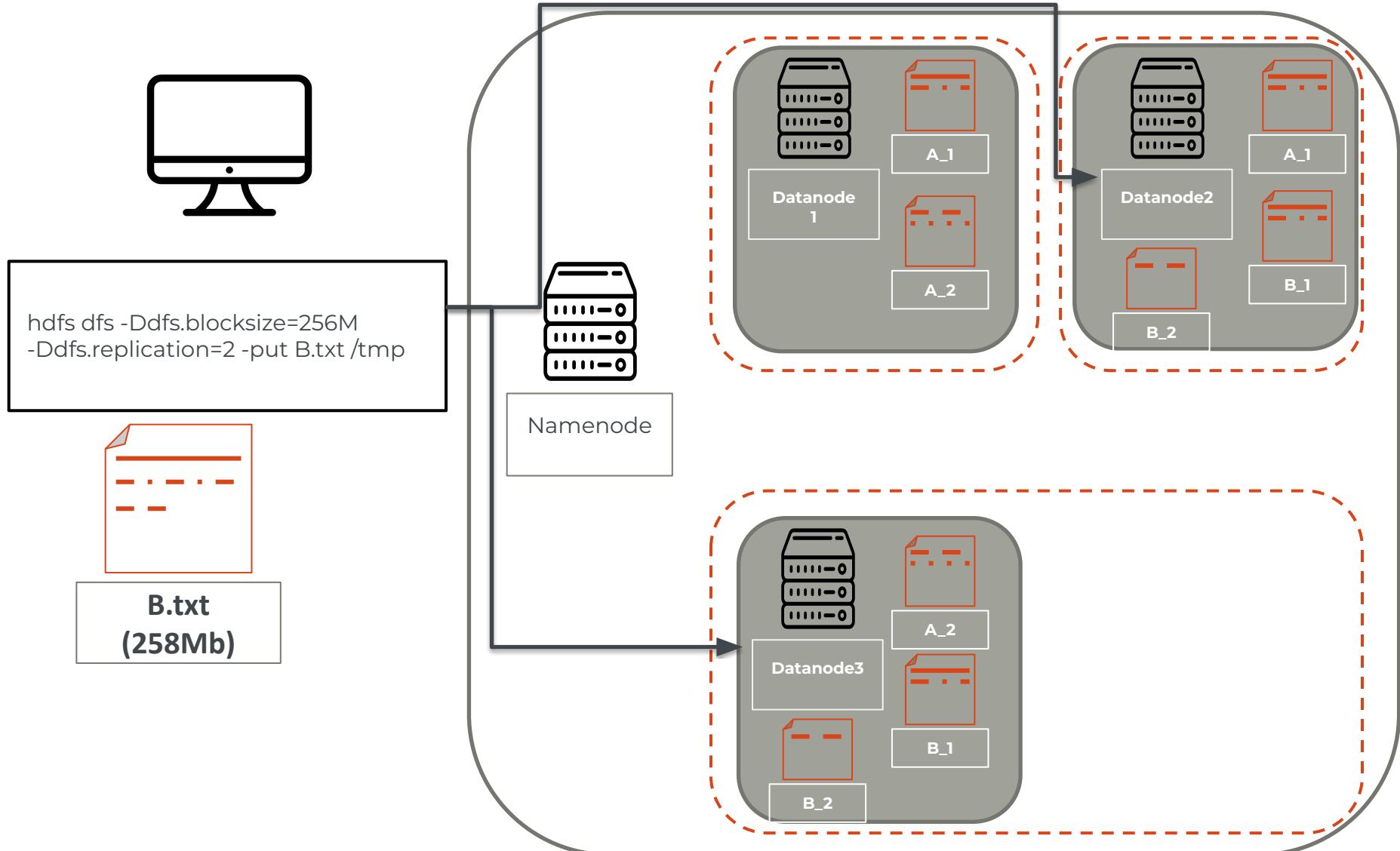
HDFS - Funcionamiento



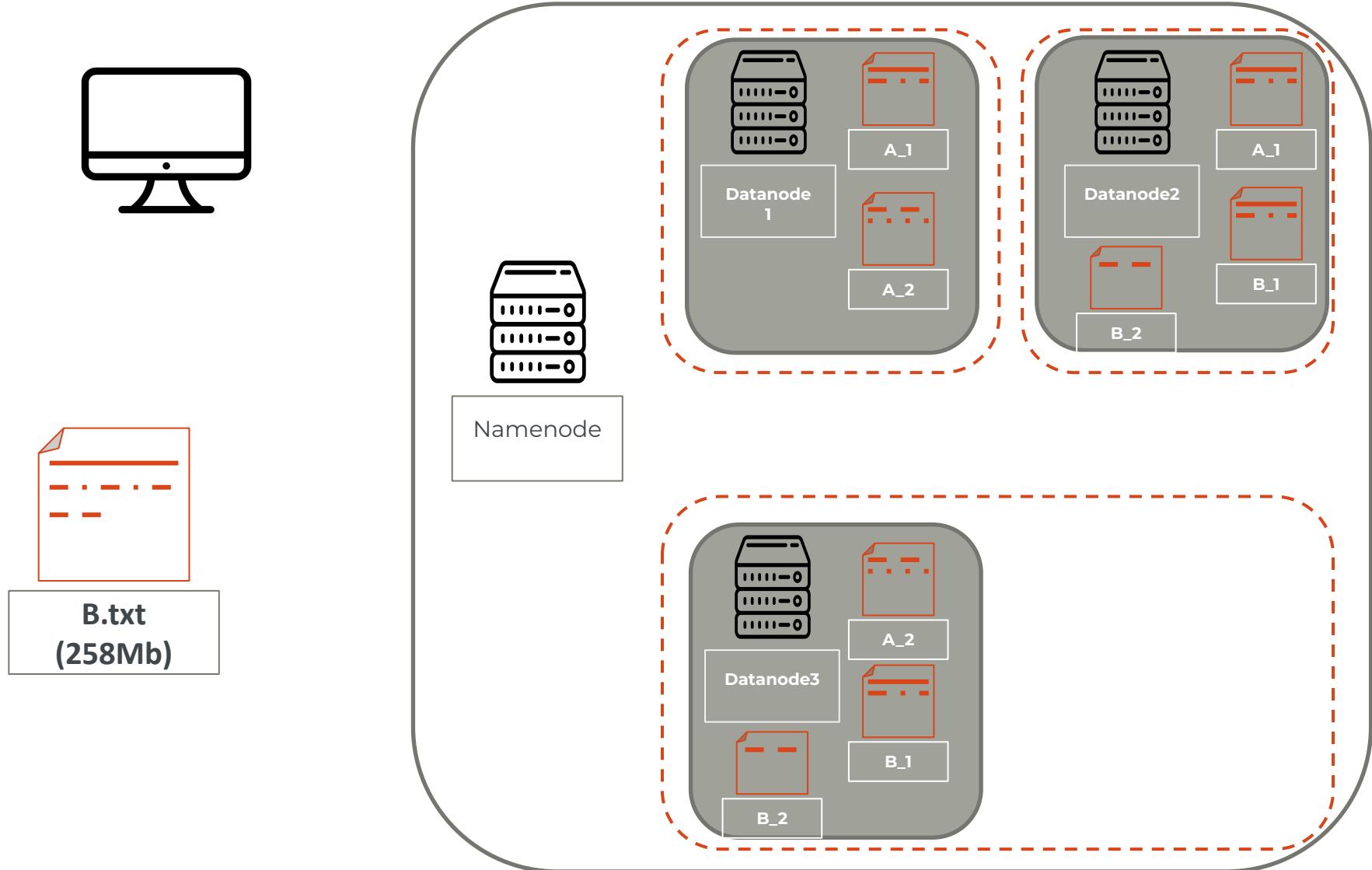
HDFS - Funcionamiento



HDFS - Funcionamiento



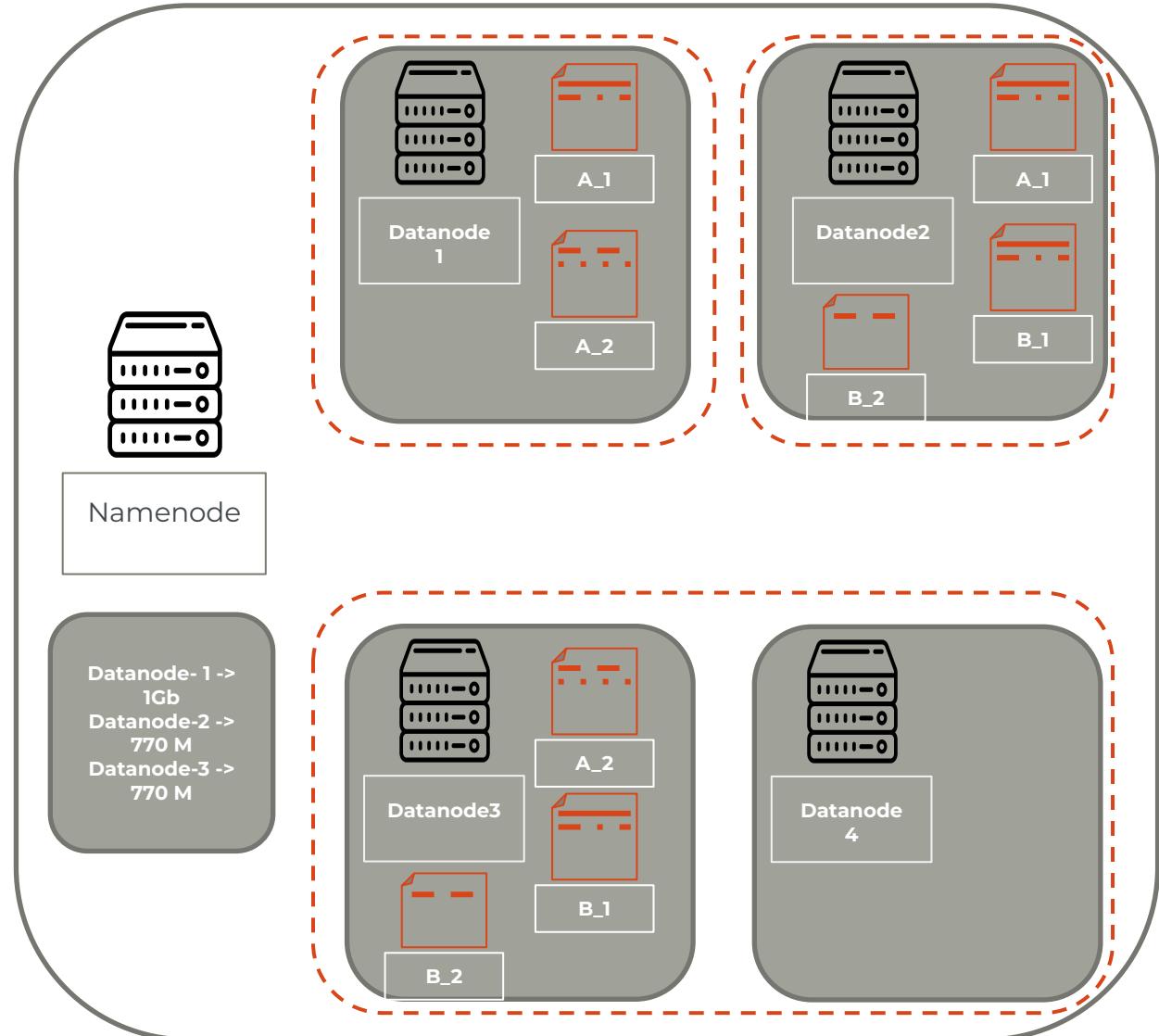
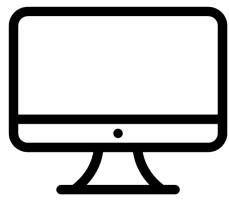
HDFS - Funcionamiento



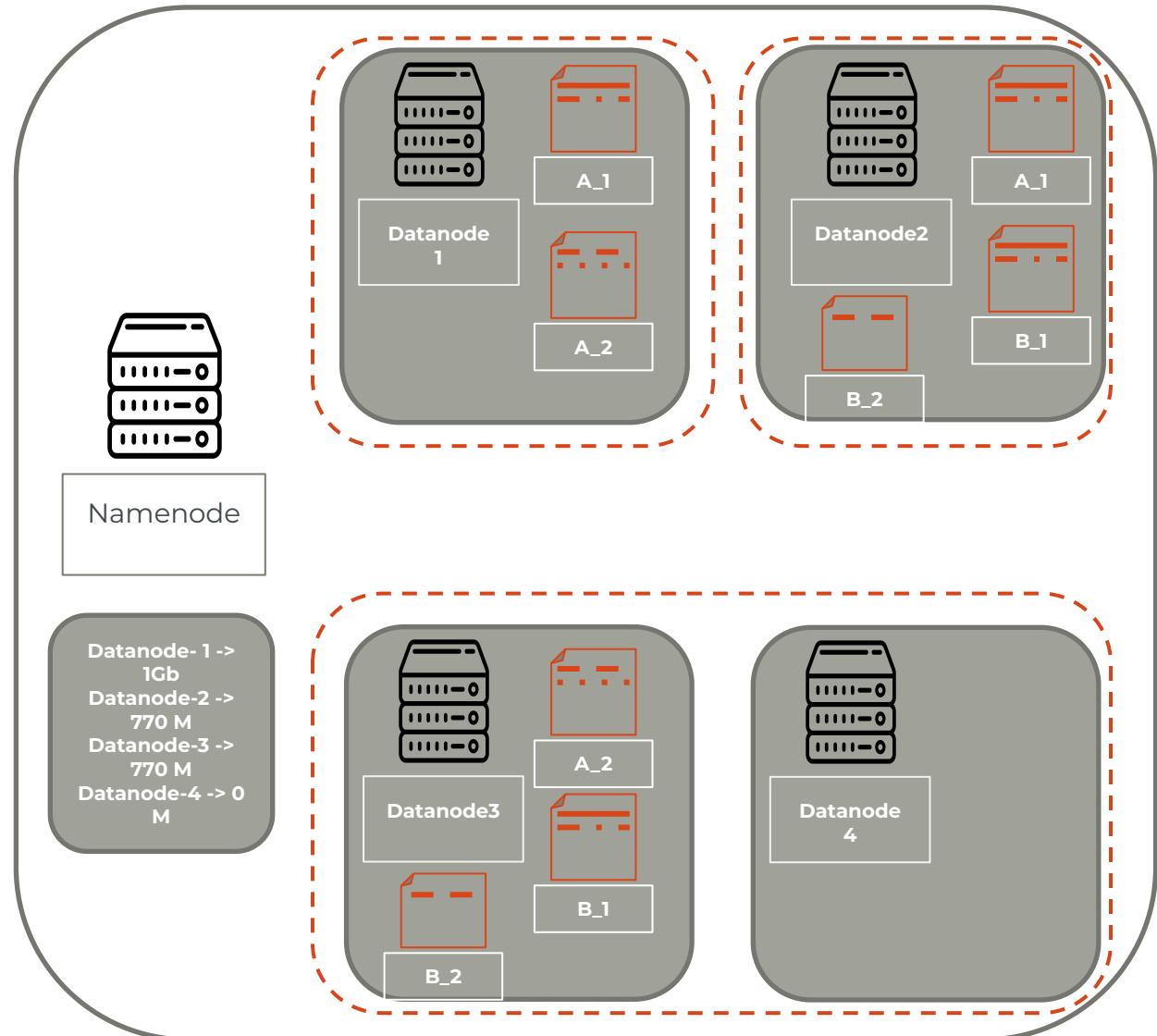
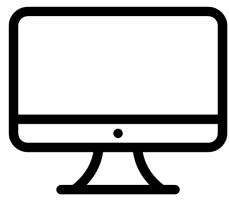
HDFS - Rebalanceamiento

- **HDFS** es una solución **horizontalmente escalable**
- Cuando se quiera aumentar la capacidad del cluster basta con añadir un nuevo **DataNode**
- A partir de ese momento se tendrá en cuenta el nuevo **DataNode** para la repartición del espacio en disco.
- Dicho DataNode, como todos los del cluster, tiene que tener conectividad con todos los nodos del cluster y con los clientes

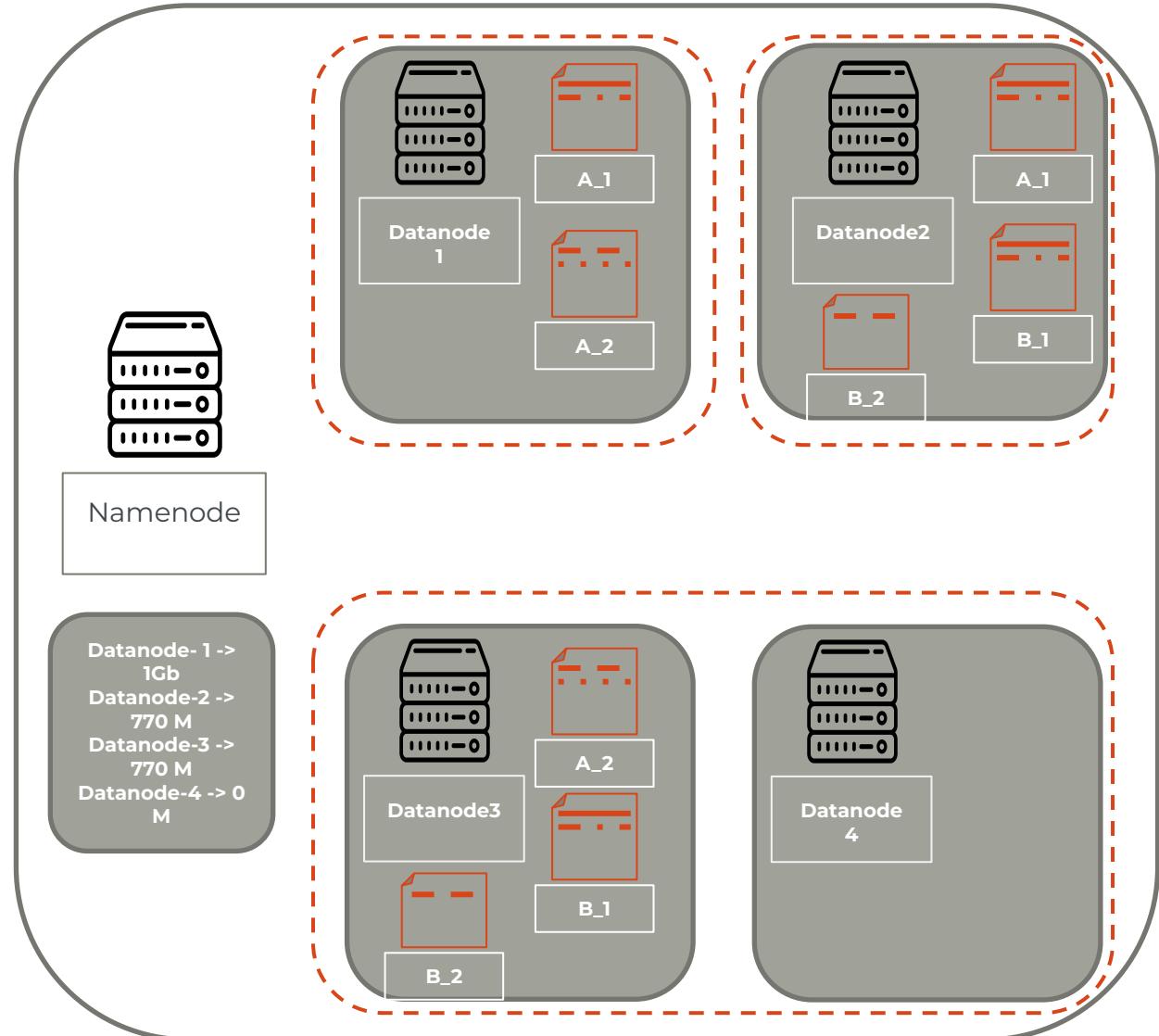
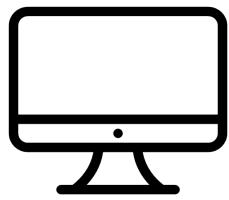
HDFS - Rebalanceamiento



HDFS - Rebalanceamiento



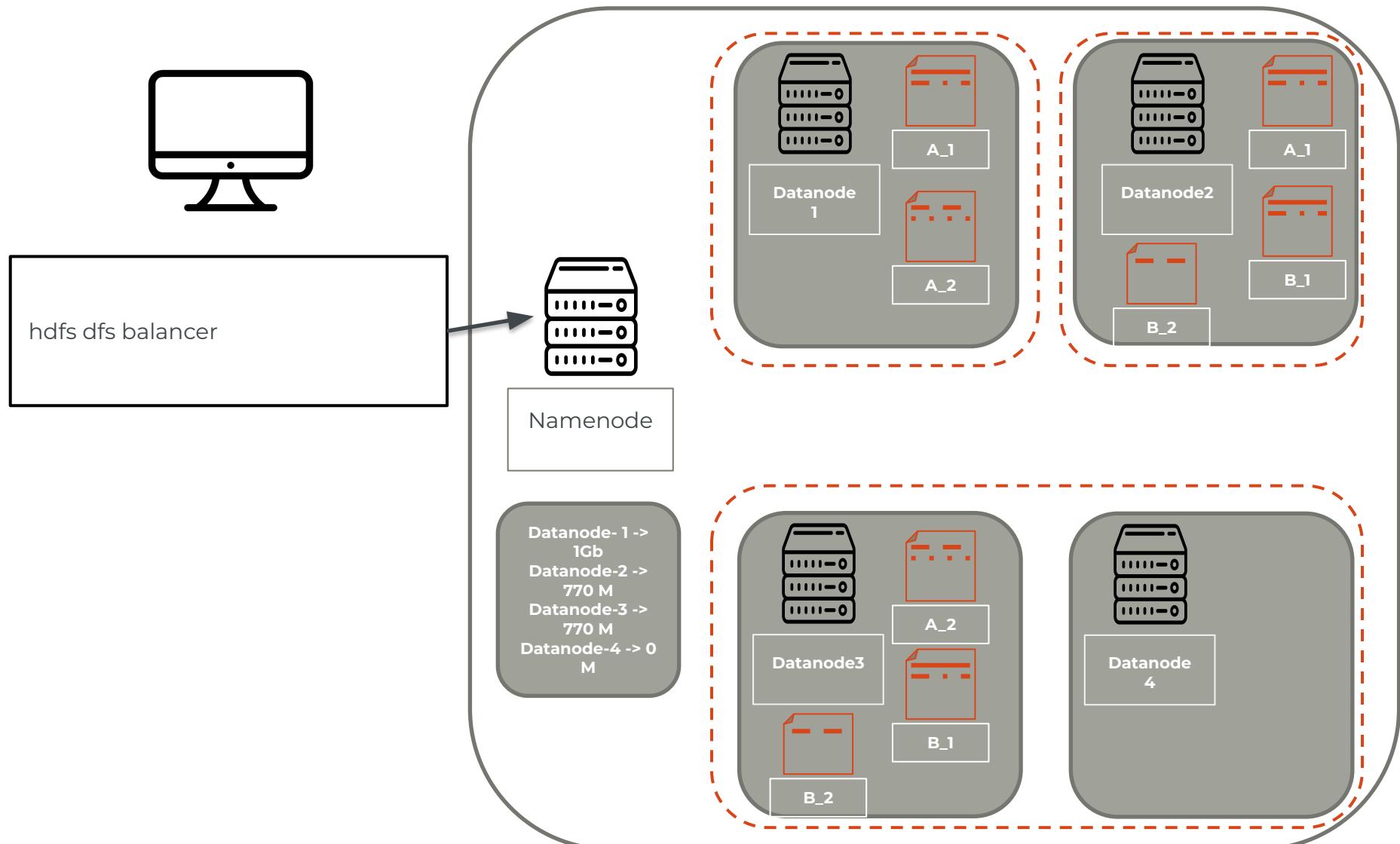
HDFS - Rebalanceamiento



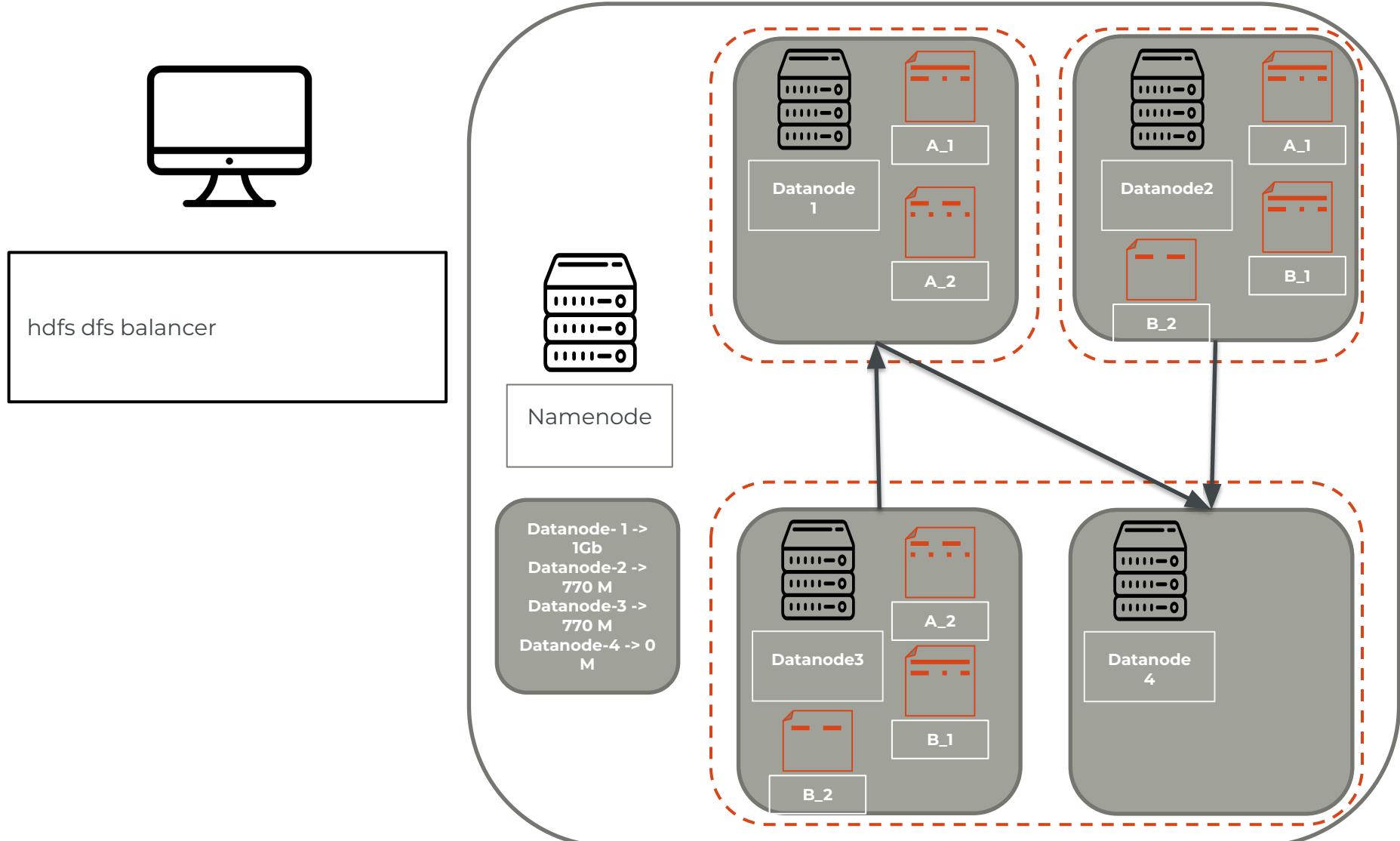
HDFS - Rebalanceamiento

- Para el correcto funcionamiento de HDFS es necesario que los nodos de datos estén balanceados
- Por defecto HDFS intentará balancear la carga por sí mismo
- Cuando esto no sea posible y los nodos se queden desbalanceados HDFS tiene un modo de balancearse
- El balanceo siempre respetará el tamaño de los bloques
- Es una operación costosa pero necesaria

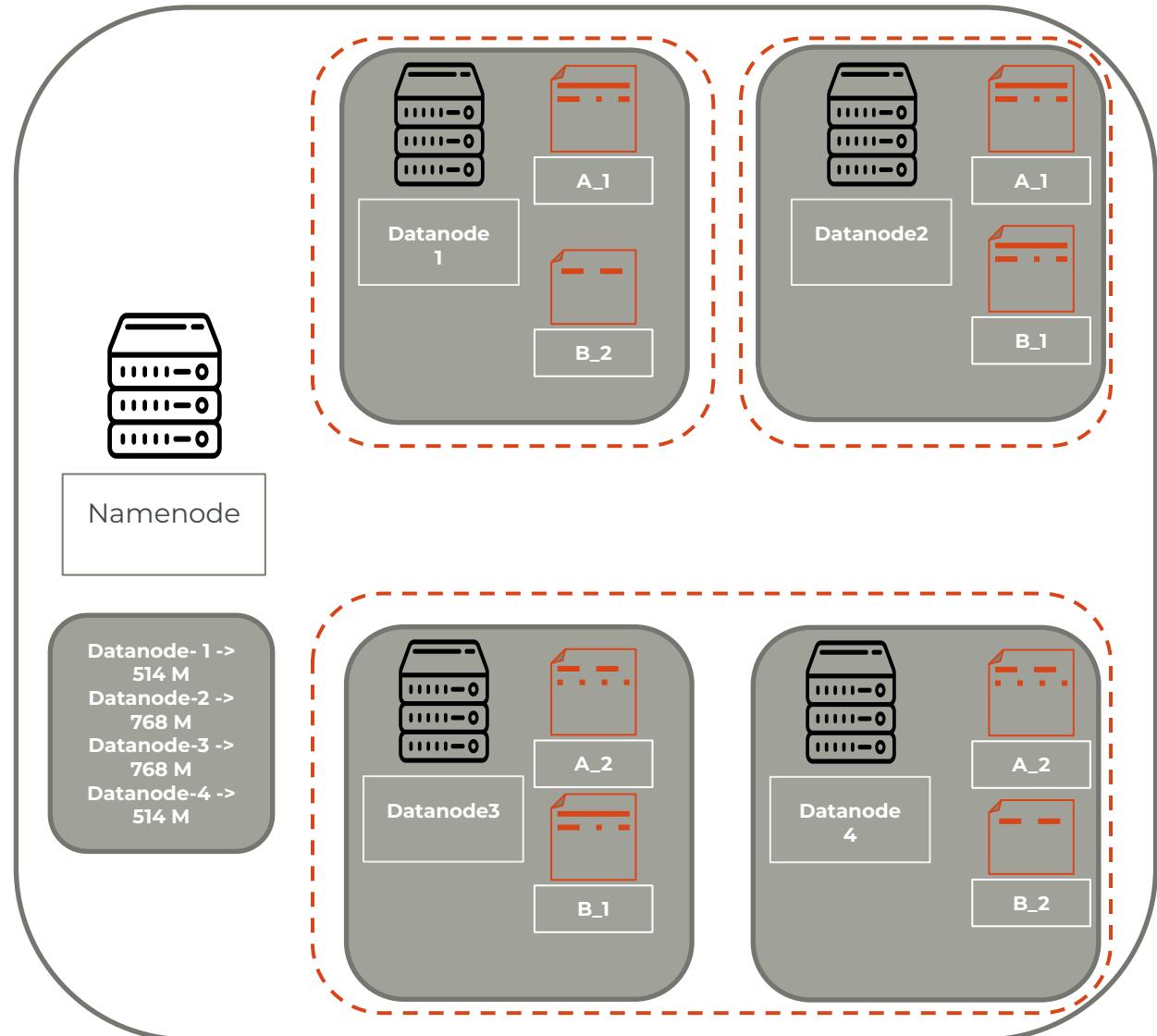
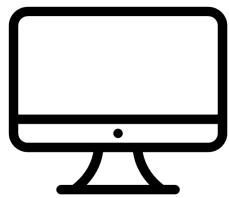
HDFS - Rebalanceamiento



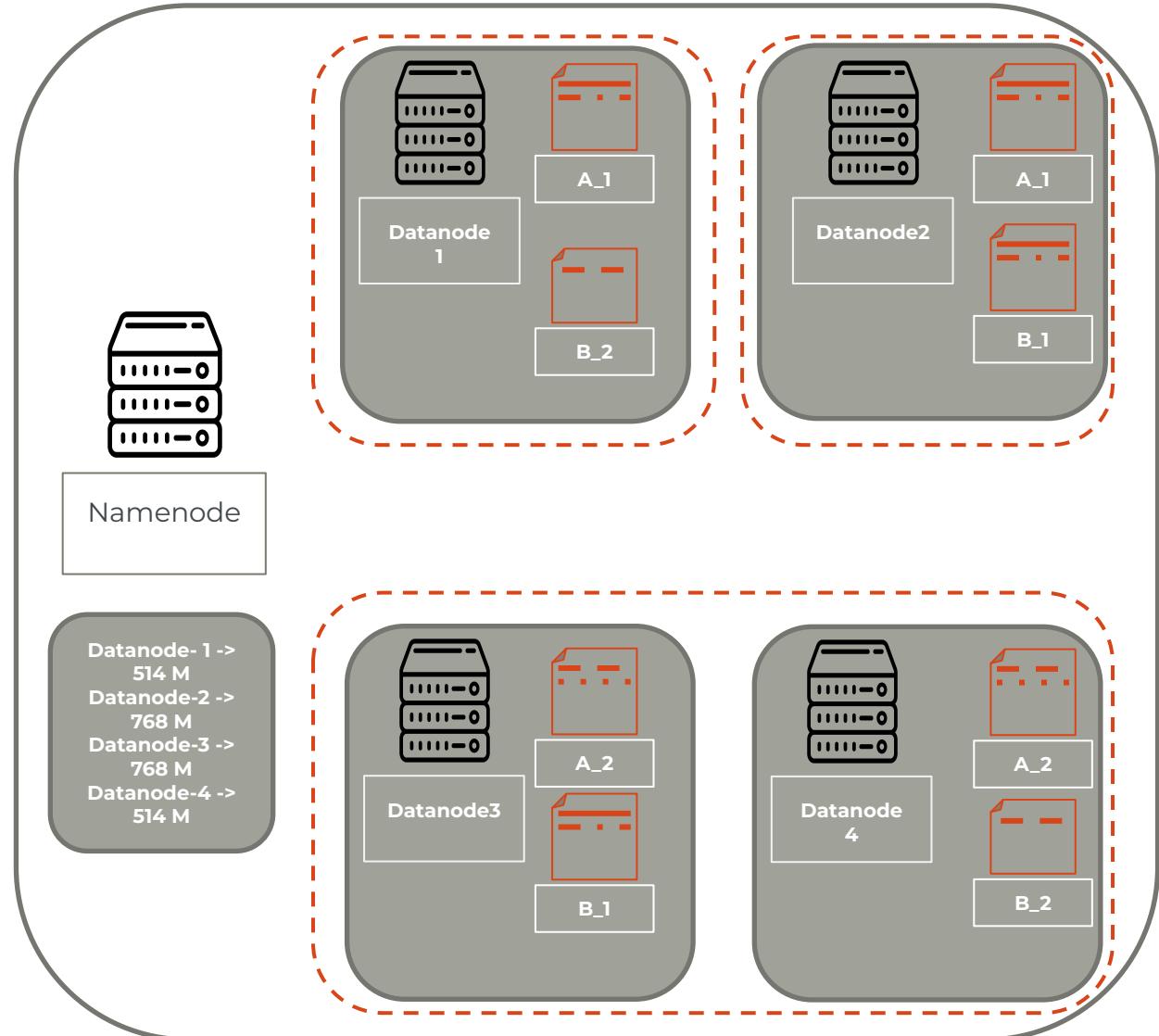
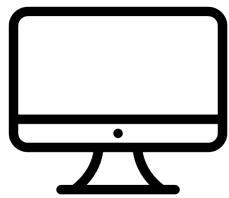
HDFS - Rebalanceamiento



HDFS - Rebalanceamiento



HDFS - Rebalanceamiento

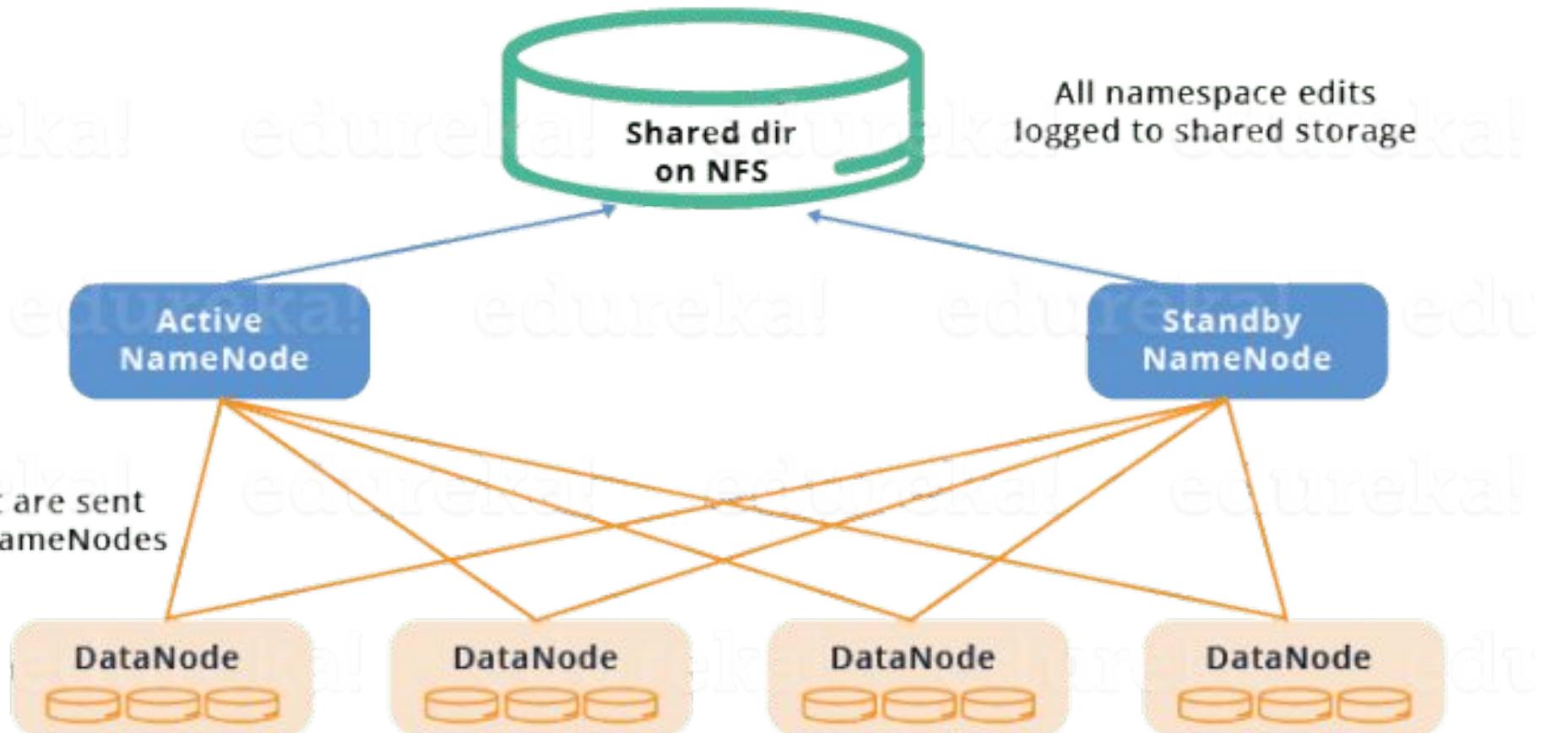


HDFS - HA

- La metainformación del cluster de HDFS se haya en el Namenode
- Todo las operaciones de HDFS tienen que pasar por el Namenode
- Aun con la réplica de bloques, el Namenode de HDFS es un punto de fallo
- HDFS sigue una estructura dos Namenodes (activo/en espera) para el HA
- HDFS en HA necesita tecnologías externas
 - Sistemas de disco en red
 - Zookeeper

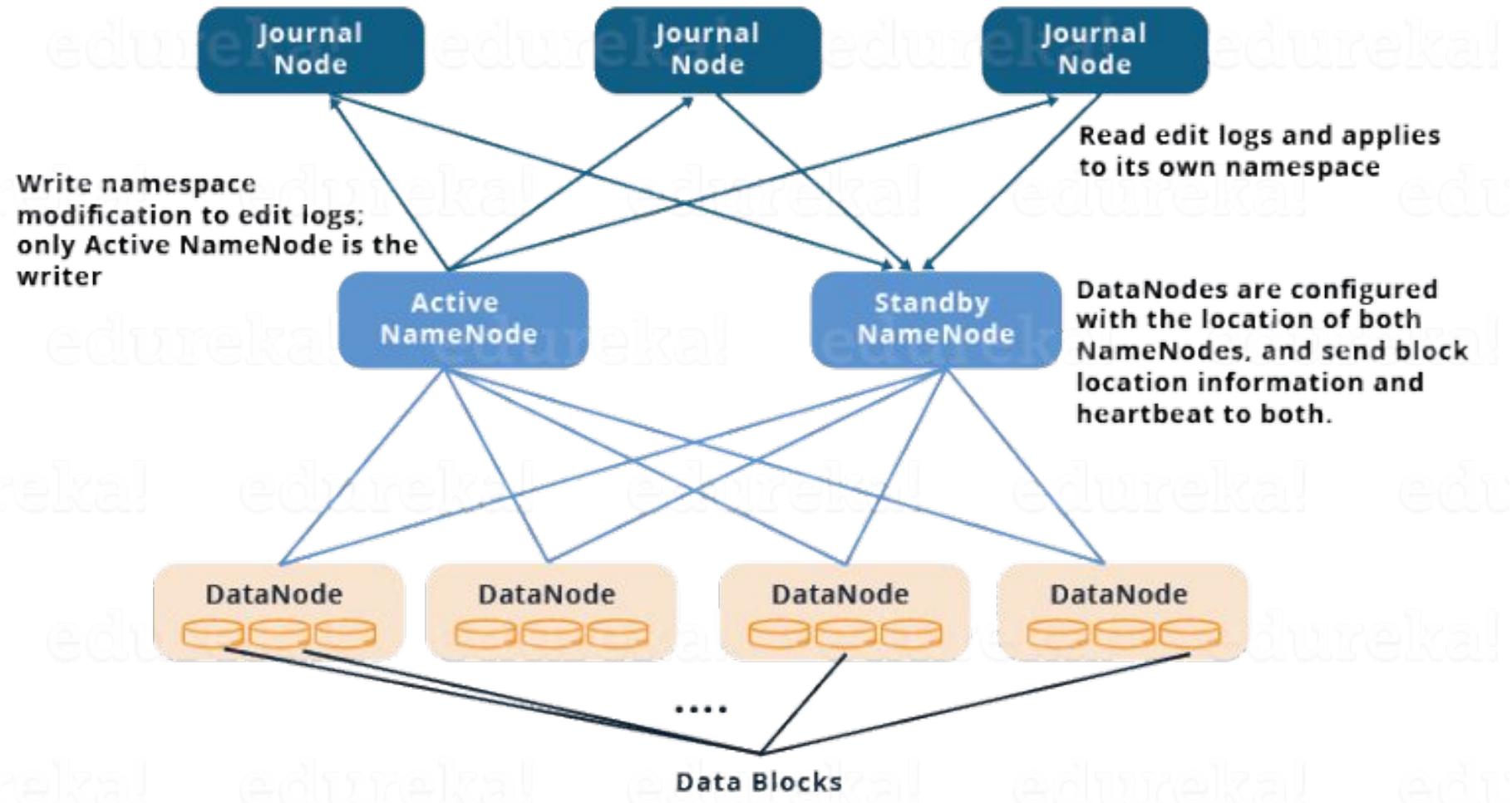
HDFS - HA

Sistemas de archivos de red

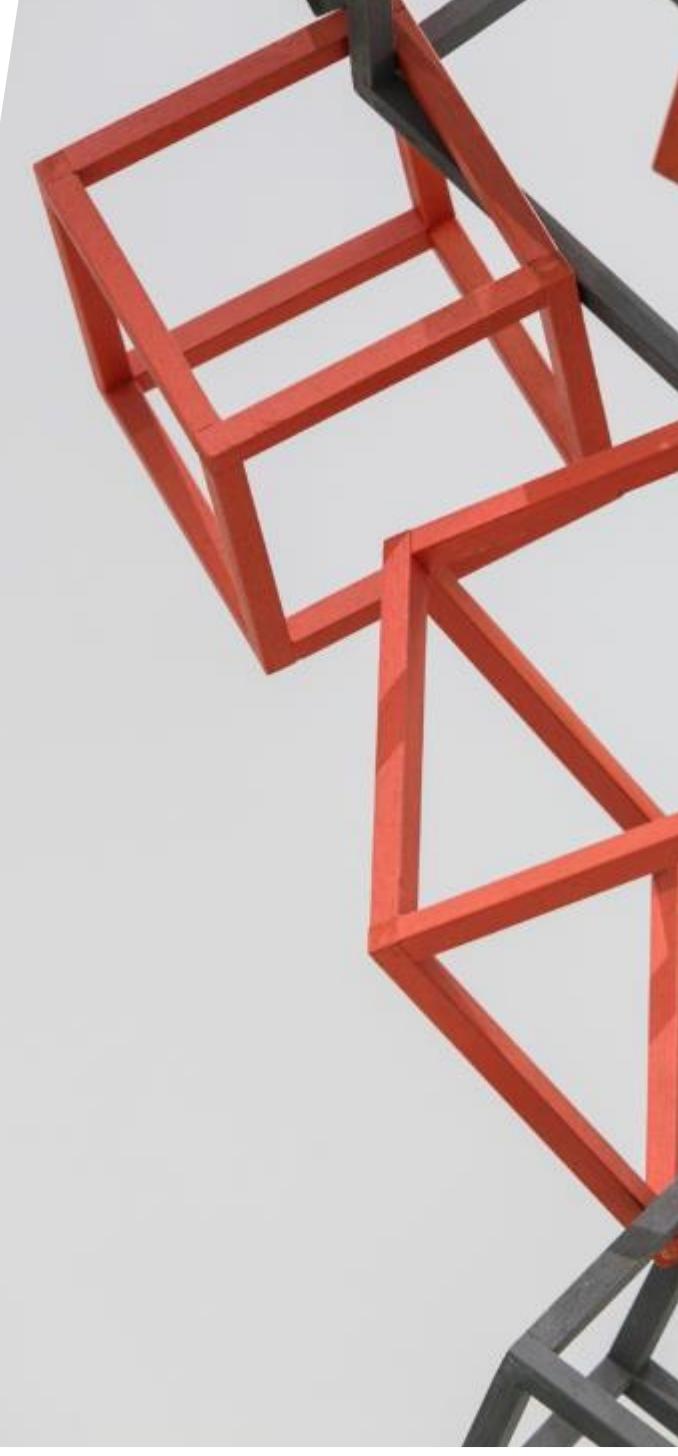


HDFS - HA

Quorum usando Journal Nodes



Apache Map&Reduce



Map & Reduce

- Modelo de programación utilizado por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras y al commodity computing.
- El nombre del framework está inspirado en los nombres de dos importantes métodos, macros o funciones en programación funcional: Map y Reduce.
- MapReduce ha sido adoptado mundialmente, ya que existe una implementación Open Source denominada Hadoop.
- Su desarrollo fue liderado inicialmente por Yahoo y actualmente lo realiza el proyecto Apache.

Map & Reduce

- Su modelo se basa en dos fases principales con una fase intermedia
 - **Map:** Fase principal en la que se transforma cada una de las unidades mínimas de cómputo para poder ser agregadas
 - **Reduce:** Fase principal en la que se transforma cada uno de los grupos de datos resultantes de la fase de Map usando funciones de agregación
 - **Shuffle:** Fase intermedia en la que se mezclan y distribuyen los datos de salida de la fase de map para ser leídos en la fase de reduce

Map & Reduce

- Su funcionamiento se basa en el concepto **Divide y Vencerás**
- Como todas las tecnologías **Big Data** tiene que estar **preparada para el fallo** y la **computación distribuida**
- En un inicio sólo podía usar **HDFS** como tecnología de almacenamiento
- Aprovecha a la comunidad Open Source para hacer nuevos conectores a sistemas de almacenamiento distribuido
- Para aprovechar al máximo su rendimiento tiene que hacer lecturas y escrituras sobre tecnologías de almacenamiento distribuidas y en bloques

Map & Reduce

- En los primeros años de Hadoop **Map & Reduce** es el único sistema de computación distribuida disponible
- Proyectos como **Apache Spark** o Apache Tez han ido sustituyendo a Hadoop **Map & Reduce**
- La forma de pensar y resolver problemas de datos masivos con estas tecnologías se basa en los mismos conceptos del algoritmo **Map & Reduce**
- No todos los problemas son "mapreducibles"

Map & Reduce - Map

Map

- Se encarga de la transformación de los datos y es aplicada en paralelo para cada unidad mínima de datos leídos desde la entrada de datos.
- Esto produce una lista de pares (k2,v2) por cada llamada.

Map(k1,v1) -> list(k2,v2)

Map & Reduce - Map

En un lugar de la mancha
de cuyo nombre
no quiero acordarme

no ha mucho tiempo
que vivía un hidalgo de los
de lanza en astillero, adarga

antigua, rocín flaco y
galgo corredor

Map(k1,v1) -> list(k2,v2)

Separar palabras,
convertir a
minúsculas y añadir
contador

(en,1)(un,1)(lugar,1)(de,1)(la,1)
(mancha,1)

(de,1)(cuyo,1)(nombre,1)

(no,1)(quiero,1)(acordarme,1)

(no,1)(ha,1)(mucho,1)(tiempo,1)

(que,1)((vivía,1)(un,1)(hidalgo,1)

(de,1)(los,1)

(de,1)(lanza,1)(en,1)(astillero,1)

(adarga,1)

(antigua,1)(rocín,1)(flaco,1)(y,1)

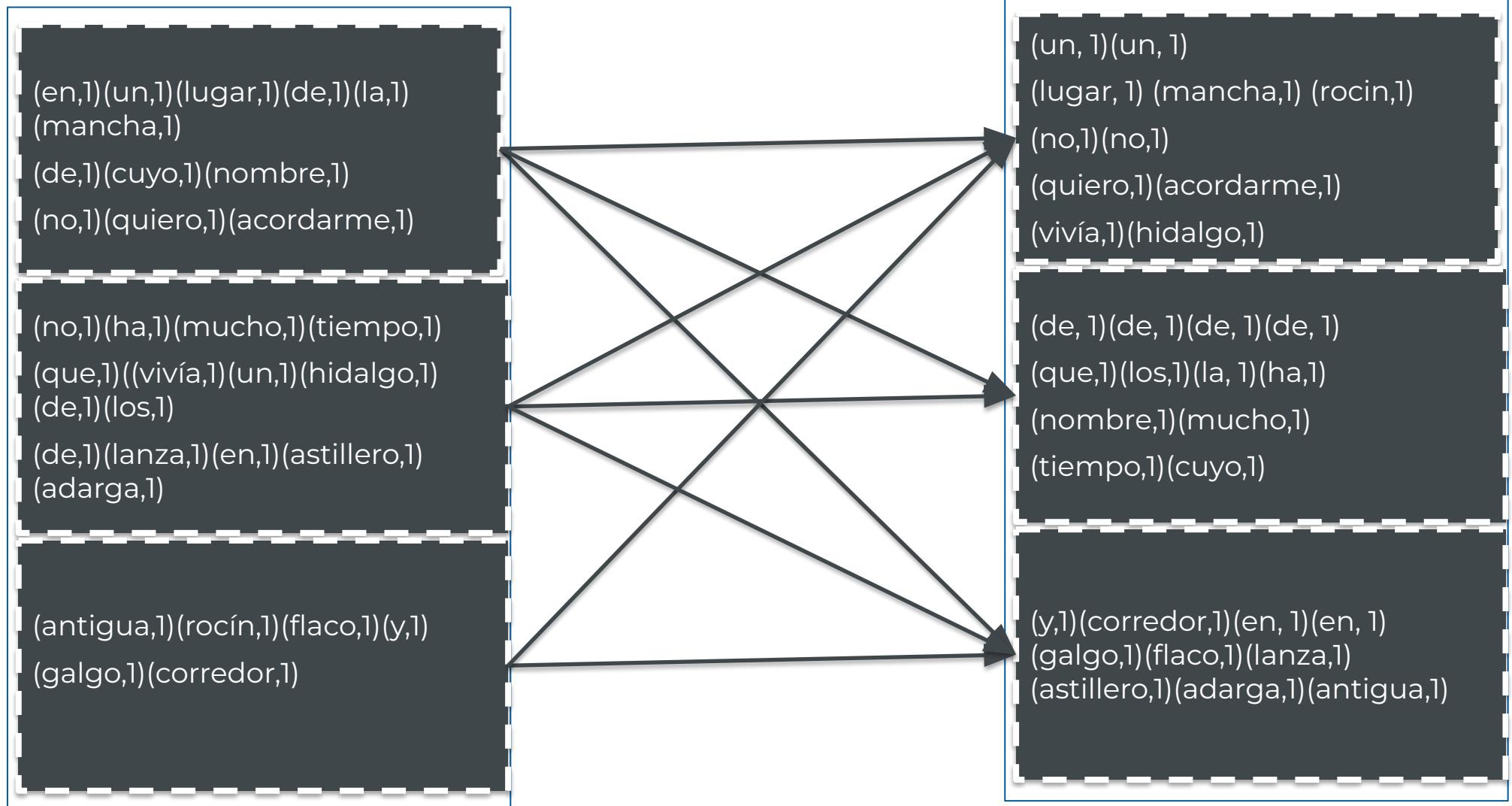
(galgo,1)(corredor,1)

Map & Reduce - Shuffle

SHUFFLE

- Se encarga de la distribución de los datos por **clave**
- Su objetivo es que **todas los valores de una misma clave estén en la memoria de un mismo nodo**
- Es la **parte más compleja del algoritmo** y se encarga de realizarla la tecnología no el desarrollador
- El desarrollador "únicamente" tiene que pensar en un conjunto de claves para que la solución sea horizontalmente escalable

Map & Reduce - Shuffle



Map & Reduce - Reduce

REDUCE

- Se encarga de la agregación de los datos y es aplicada en paralelo para lista de datos(v2) con la misma clave (k2) resultante de la fase de map
- Esto produce una lista elementos (v3) por cada lista de elementos.

Reduce: (k2,list(v2)) -> list(v3)

Map & Reduce - Reduce

```
(un, 1)(un, 1)  
(lugar, 1) (mancha,1) (rocín,1)  
(no,1)(no,1)  
(quiero,1)(acordarme,1)  
(vivía,1)(hidalgo,1)
```

```
(de, 1)(de, 1)(de, 1)(de, 1)  
(que,1)(los,1)(la, 1)(ha,1)  
(nombre,1)(mucho,1)  
(tiempo,1)(cuyo,1)
```

```
(y,1)(corredor,1)(en, 1)(en, 1)  
(galgo,1)(flaco,1)(lanza,1)  
(astillero,1)(adarga,1)(antigua,1)
```

Sumar los contadores

```
(un, 2)(lugar, 1)  
(mancha,1) (rocín,1)  
(no,2)(quiero,1)  
(acordarme,1)(vivía,1)(hidalgo,1)
```

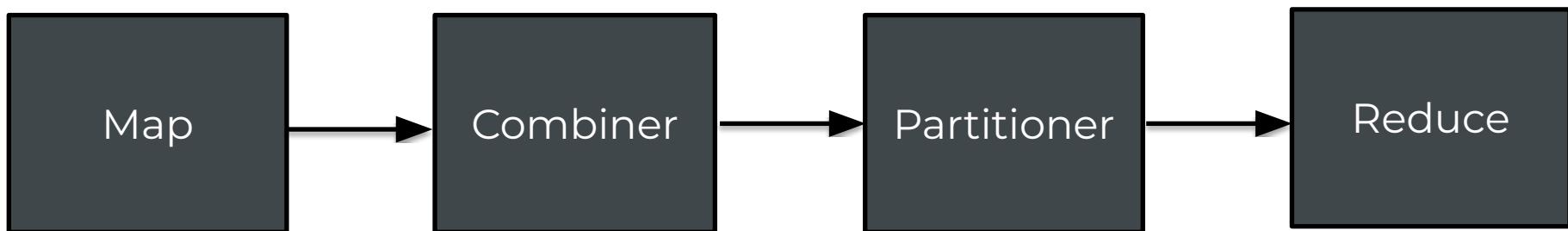
```
(de, 4)(que,1)(los,1)(la, 1)  
(ha,1)(nombre,1)(mucho,1)  
(tiempo,1)(cuyo,1)
```

```
(y,1)(corredor,1)(en, 1)(en, 1)  
(galgo,1)(flaco,1)(lanza,1)  
(astillero,1)(adarga,1)(antigua,1)
```

Map & Reduce - Fases Opcionales

- **Map, Reduce** y **Shuffle** son las fases del algoritmo Map & Reduce
- Cuando se aplicó este algoritmo en para la resolución de problemas de datos masivos se vio que se podían añadir fases intermedias
- Estas fases sólo mejoran el rendimiento
- No están pensadas para solucionar problemas no tratables con **Map** y **Reduce**
- Las fases son:
 - *Combiner*
 - *Partitioner*

Map & Reduce - Fases Opcionales



Map & Reduce - Combiner

COMBINER

- **Map & Reduce** se basa en un algoritmo matemático
- Cantidades masivas de datos requieren mucho tráfico de red
- Hadoop se funda en la base de "*el almacenamiento es gratis*"
- Cuando se crea Hadoop el cuello de botella es el tráfico de red
- Se añade una fase nueva entre el **Map** y el **Shuffle** llamada **Combiner**
- Se basa en un "*mini reduce*"

Map & Reduce - Combiner

En un lugar de la mancha
de cuyo nombre
no quiero acordarme

no ha mucho tiempo que vivía
un hidalgo de los de lanza en
astillero, adarga

antigua, rocín flaco y
galgo corredor

(en,1)(un,1)(lugar,1)(de,1)(la,1)
(mancha,1)
(de,1)(cuyo,1)(nombre,1)
(no,1)(quiero,1)(acordarme,1)

(no,1)(ha,1)(mucho,1)(tiempo,1)
(que,1)((vivía,1)(un,1)(hidalgo,1)
(de,1)(los,1)
(de,1)(lanza,1)(en,1)(astillero,1)
(adarga,1)

(antigua,1)(rocín,1)(flaco,1)(y,1)
(galgo,1)(corredor,1)

(en,1)(un,1)(lugar,1)(la,1)
(mancha,1)
(de,2)(cuyo,1)(nombre,1)
(no,1)(quiero,1)(acordarme,1)

(no,1)(ha,1)(mucho,1)(tiempo,1)
(que,1)((vivía,1)(un,1)(hidalgo,1)(los,
1)
(de,2)(lanza,1)(en,1)(astillero,1)
(adarga,1)

(antigua,1)(rocín,1)(flaco,1)(y,1)
(galgo,1)(corredor,1)

Map & Reduce - Partitioner

PARTITIONER

- Para hacer un Reduce siempre hay que hacer un map previo
- Hay que intentar siempre reducir el tráfico de red
- Mediante el método partitioner obligamos a varias claves a ir al mismo nodo
- Se reduce el número de maps
- Se tiene que conocer la distribución de las claves de antemano

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos, calcula el número de vuelos que salen de cada aeropuerto

Origen,Destino,Fecha,Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC,BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos, calcula el número de pasajeros que llegan a cada aeropuerto

Origen,Destino,Fecha,Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC,BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos, calcula el aeropuerto con mayor número de pasajeros en Marzo

Origen,Destino,Fecha,Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC,BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos y el precio de los menús por pasajero y mes, calcula el país con mayor número de recepción de viajeros

Origen, Destino, Fecha, Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC, BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Aeropuerto, País

NYC, USA
SFO, USA
LAX, USA
MAD, SPA

BOS, USA
LAS, USA
BCN, SPA
LHR, GBR

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos, calcula la media de viajeros salientes por mes y aeropuerto

Origen,Destino,Fecha,Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC, BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos, calcula la mediana de viajeros salientes por mes y aeropuerto

Origen,Destino,Fecha,Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC, BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Map & Reduce - Ejercicios

Dado el siguiente registro de vuelos y el precio de los menús por pasajero y mes, calcula el aeropuerto con mayor número de viajeros procedentes de otro país

Origen, Destino, Fecha, Pasajeros

NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC, BOS,20200203,100
MAD,BCN,20200301,120

NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215

LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108

Aeropuerto, País

NYC, USA
SFO, USA
LAX, USA
MAD, SPA

BOS, USA
LAS, USA
BCN, SPA
LHR, GBR

Apache YARN



YARN- Historia

- En los inicios de Hadoop sólo existía Map & Reduce como procesador
- Para gestionar sus procesos se crea Jobtracker
- Seguía una arquitectura de Maestro/Eslavo (JobTracker/TaskTracker)
- Cuando se creó no se tuvo en cuenta lo que podían e iban a crecer los cluster de Hadoop
- Se convierte en cuello de botella con cluster grandes (más de 4000 nodos)
- Sólo admite interacción con la interfaz de Map & Reduce
- Se encarga de asignar el trabajo de cada Mapper o Reducer a un nodo

YARN- Componentes

- Su arquitectura es de Maestro/Eslavo
- Permite la ejecución de algo más que Map&Reduce
- Sigue siendo "únicamente" un gestor de aplicaciones no de recursos
- Tiene 4 componentes principales
 - **ResourceManager**
 - **NodeManager**
 - **ApplicationMaster**
 - **Container**

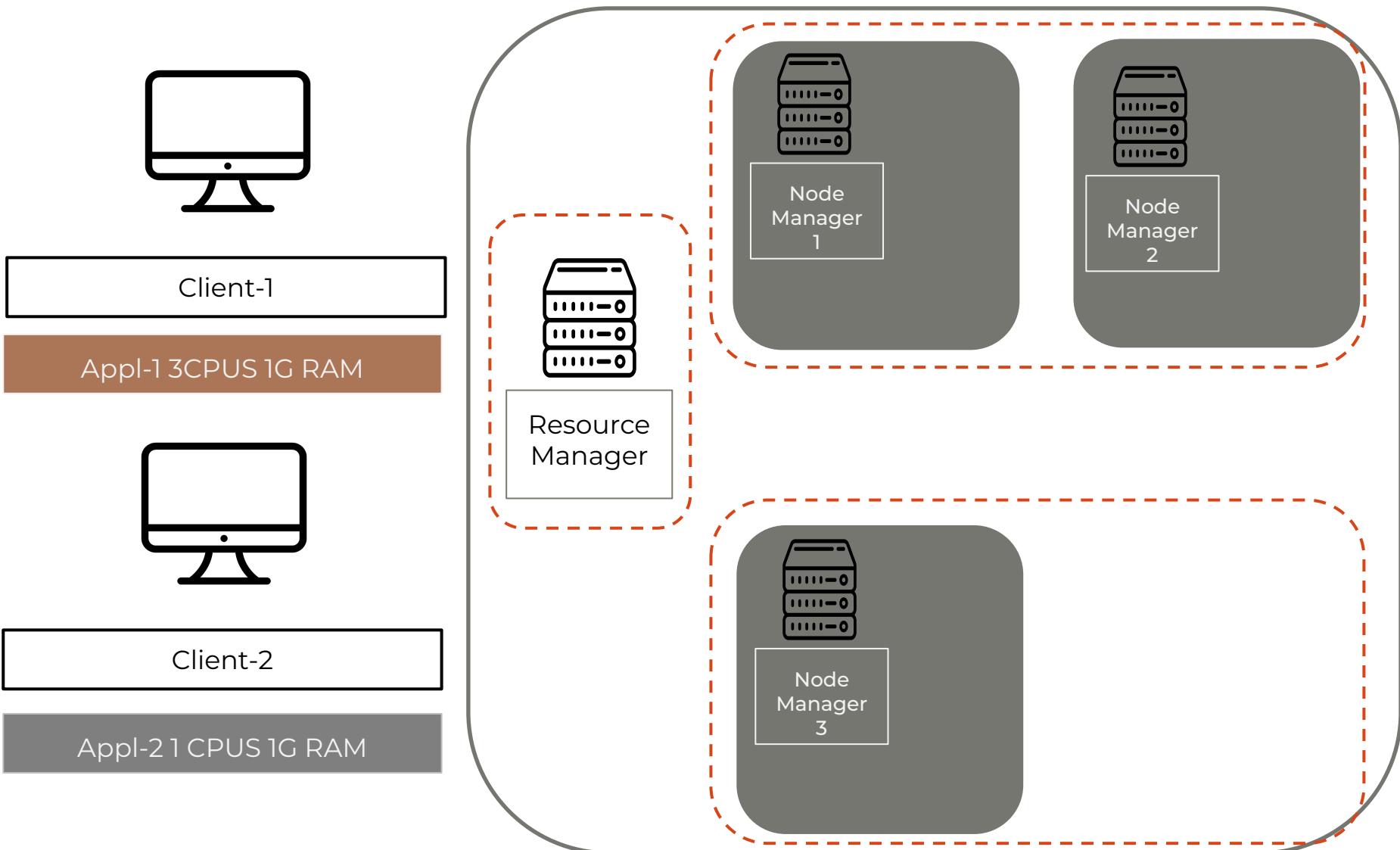
YARN- Componentes

- **ResourceManager**
 - Planifica las ejecuciones asignando los recursos de los nodos(Scheduler) y monitoriza el estado de las aplicaciones(ApplicationMaster)
- **NodeManager:**
 - Gestiona las tareas asignadas por el Resource Manager
- **ApplicationMaster**
 - Coordina la ejecución de las aplicaciones dentro del cluster
- **Container**
 - Se encarga de la ejecución de cada una de las tareas de una aplicación

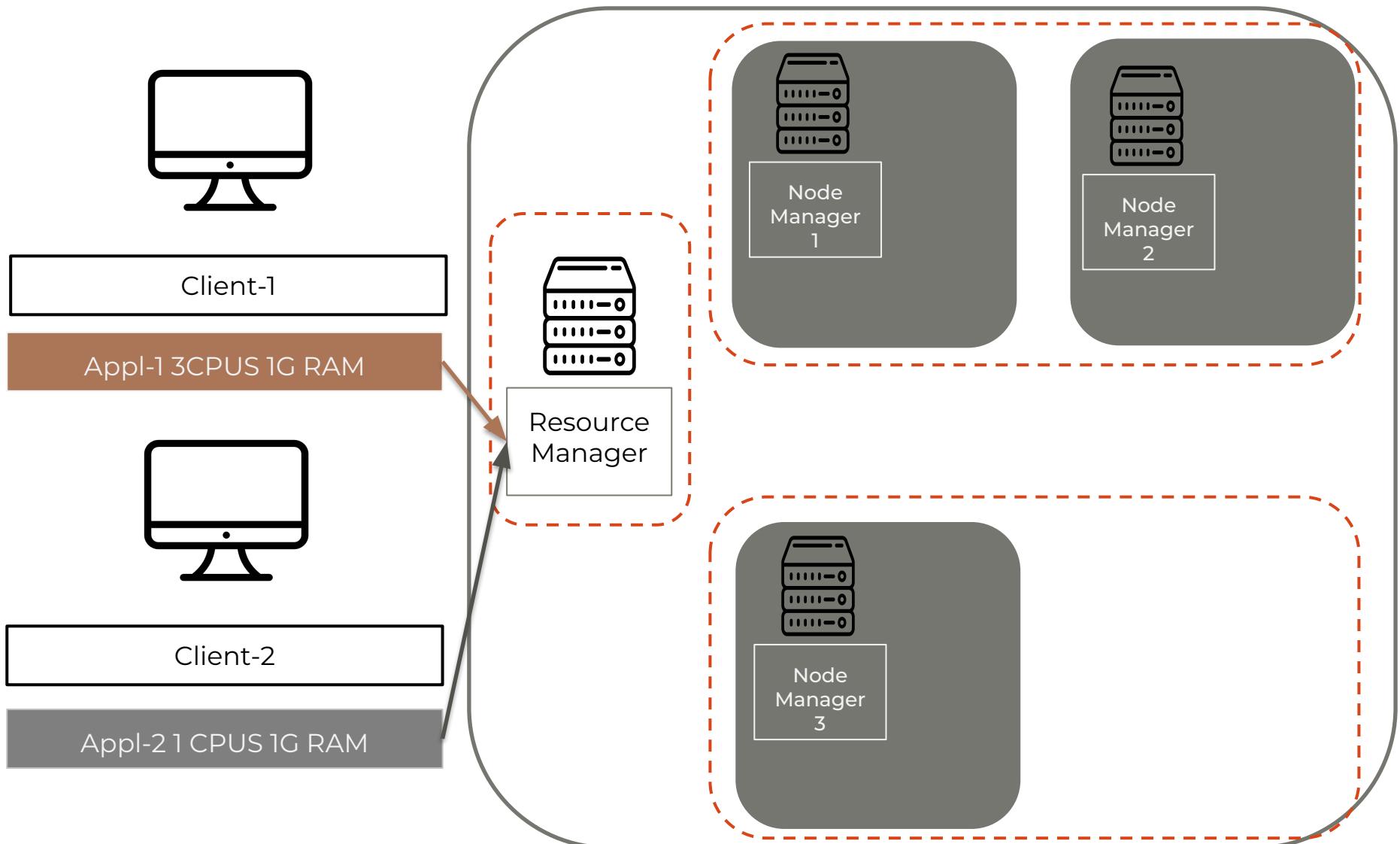
YARN- Componentes

- **ResourceManager**
 - Planifica las ejecuciones asignando los recursos de los nodos(Scheduler) y monitoriza el estado de las aplicaciones(ApplicationMaster)
- **NodeManager:**
 - Gestiona las tareas asignadas por el Resource Manager
- **ApplicationMaster**
 - Coordina la ejecución de las aplicaciones dentro del cluster
- **Container**
 - Se encarga de la ejecución de cada una de las tareas de una aplicación

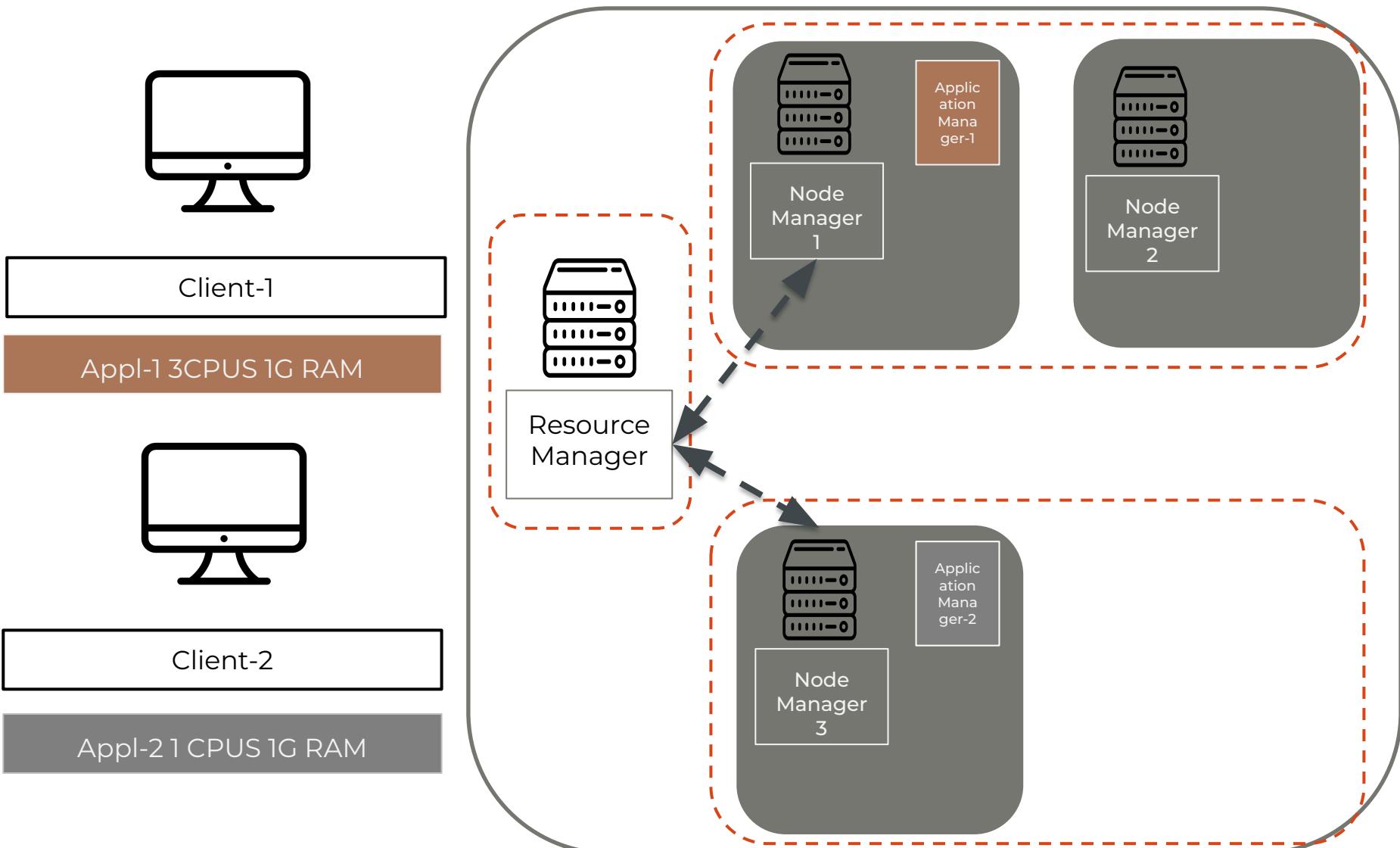
YARN - M&R Integración



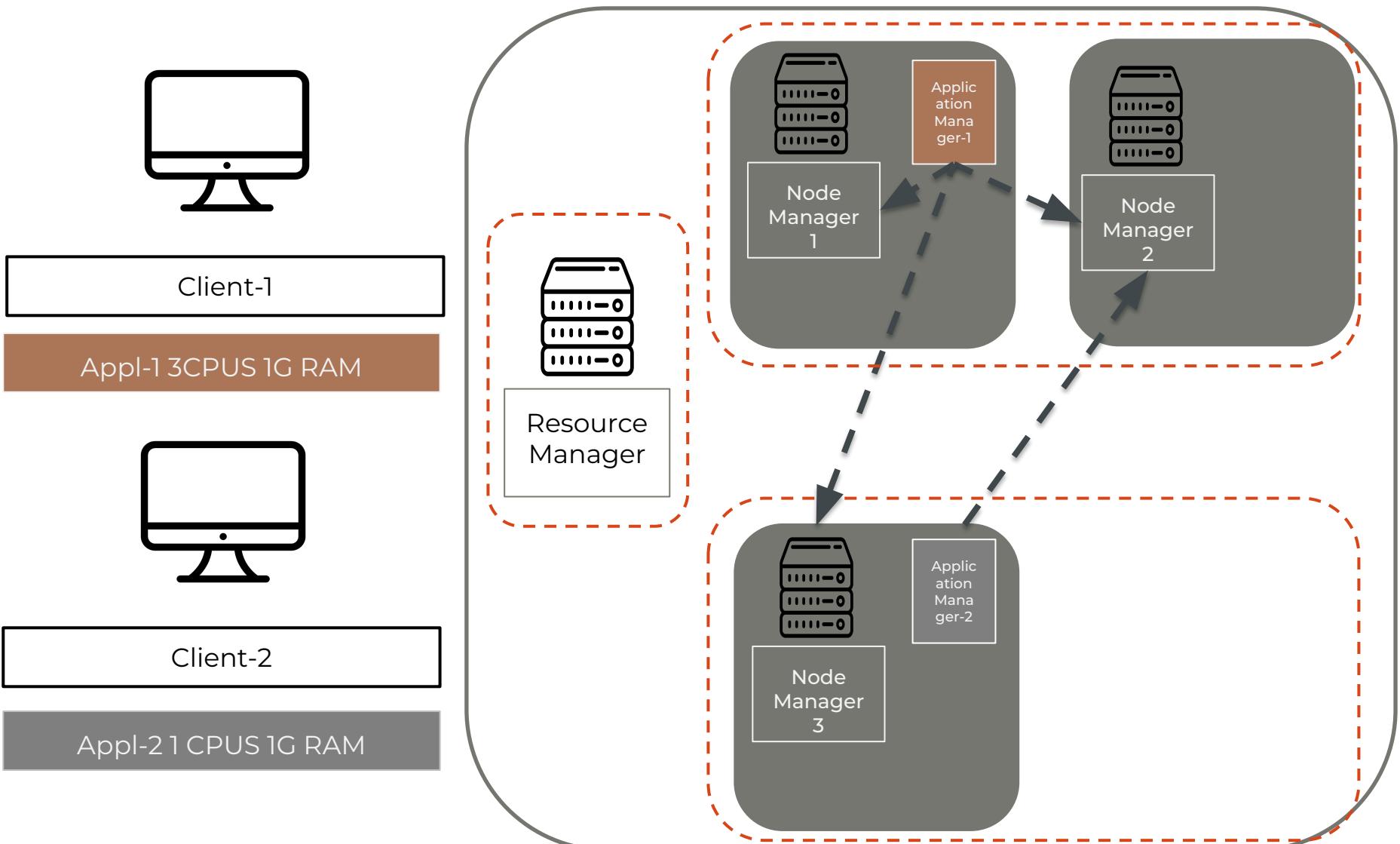
YARN - M&R Integracion



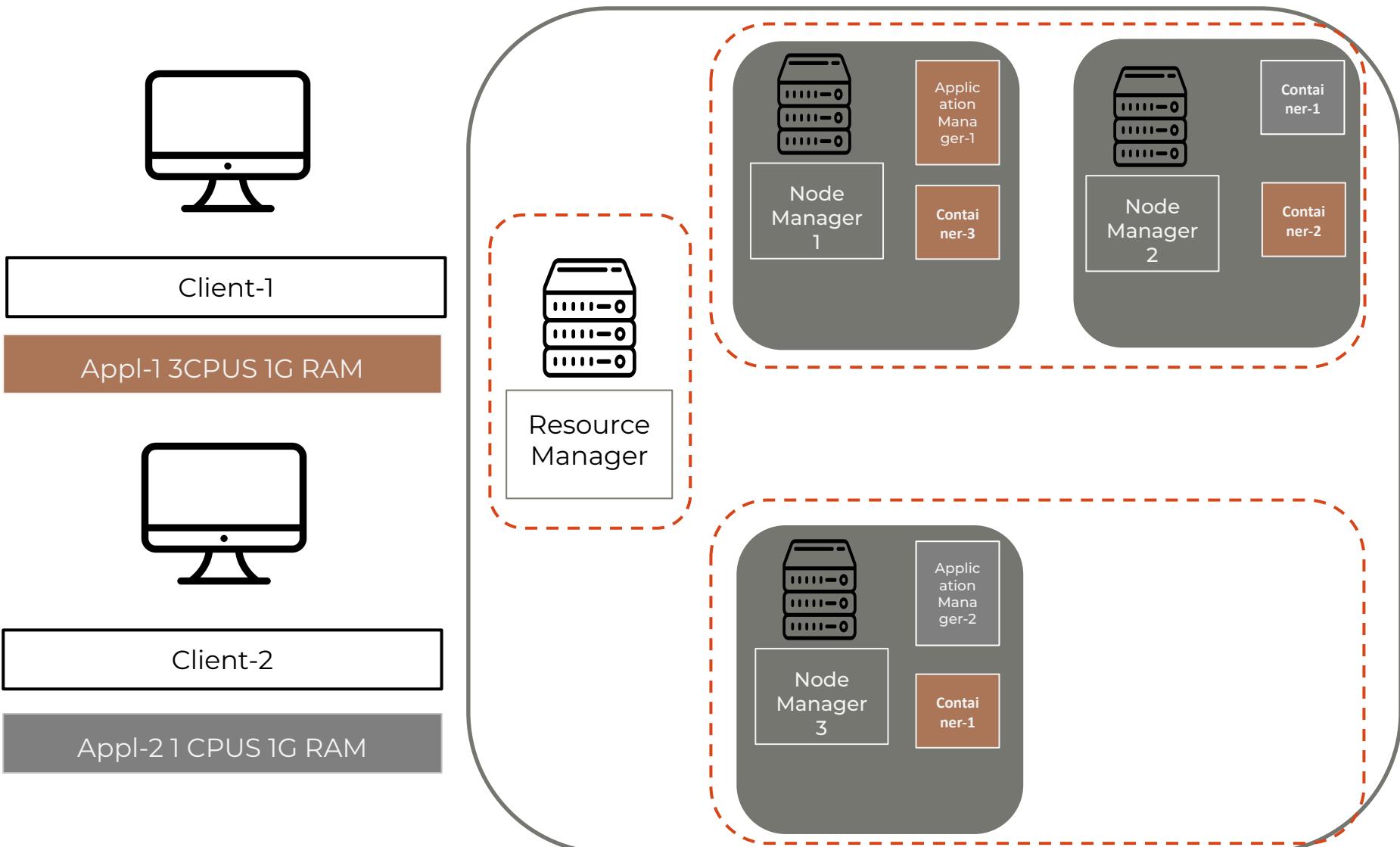
YARN - M&R Integracion



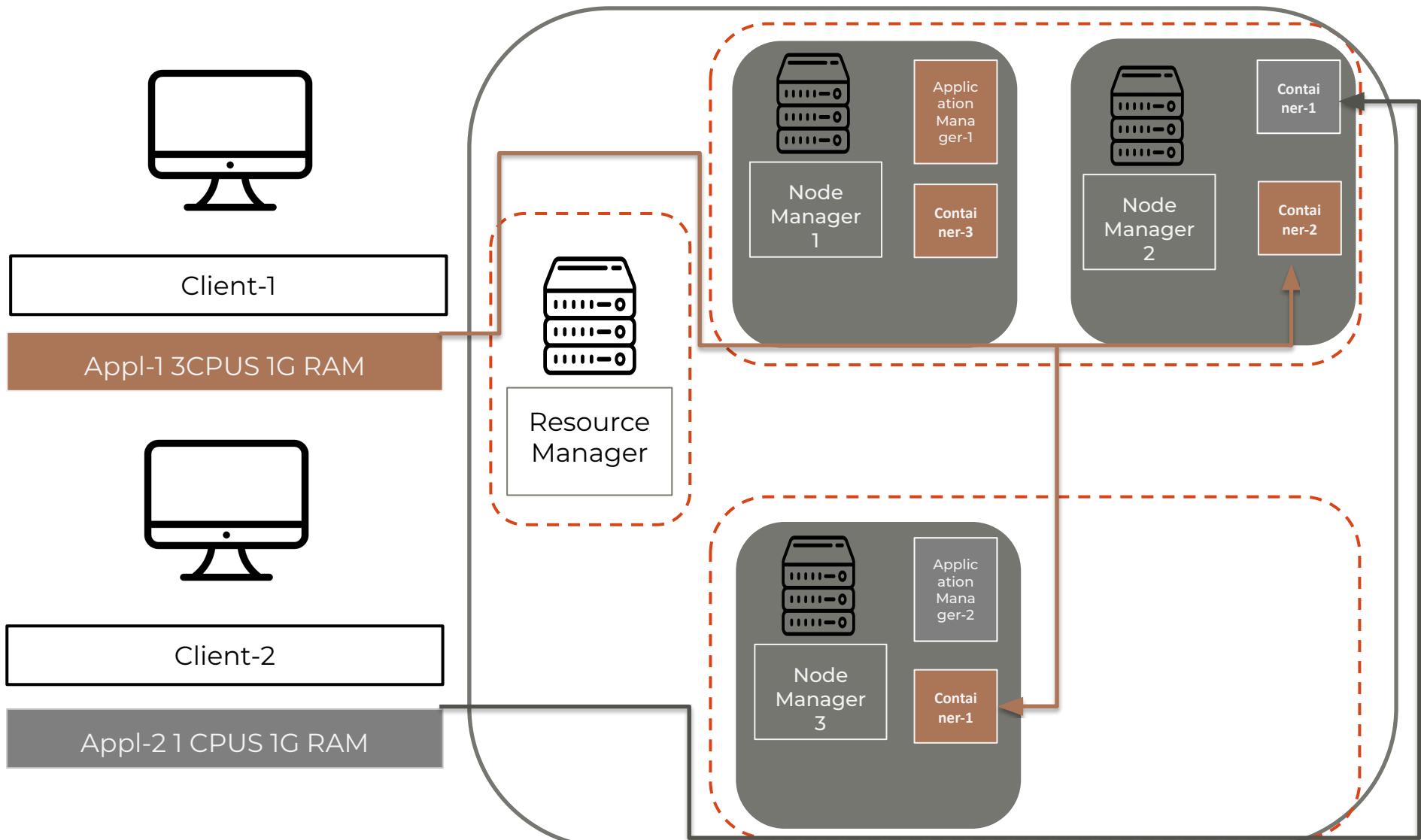
YARN - M&R Integracion



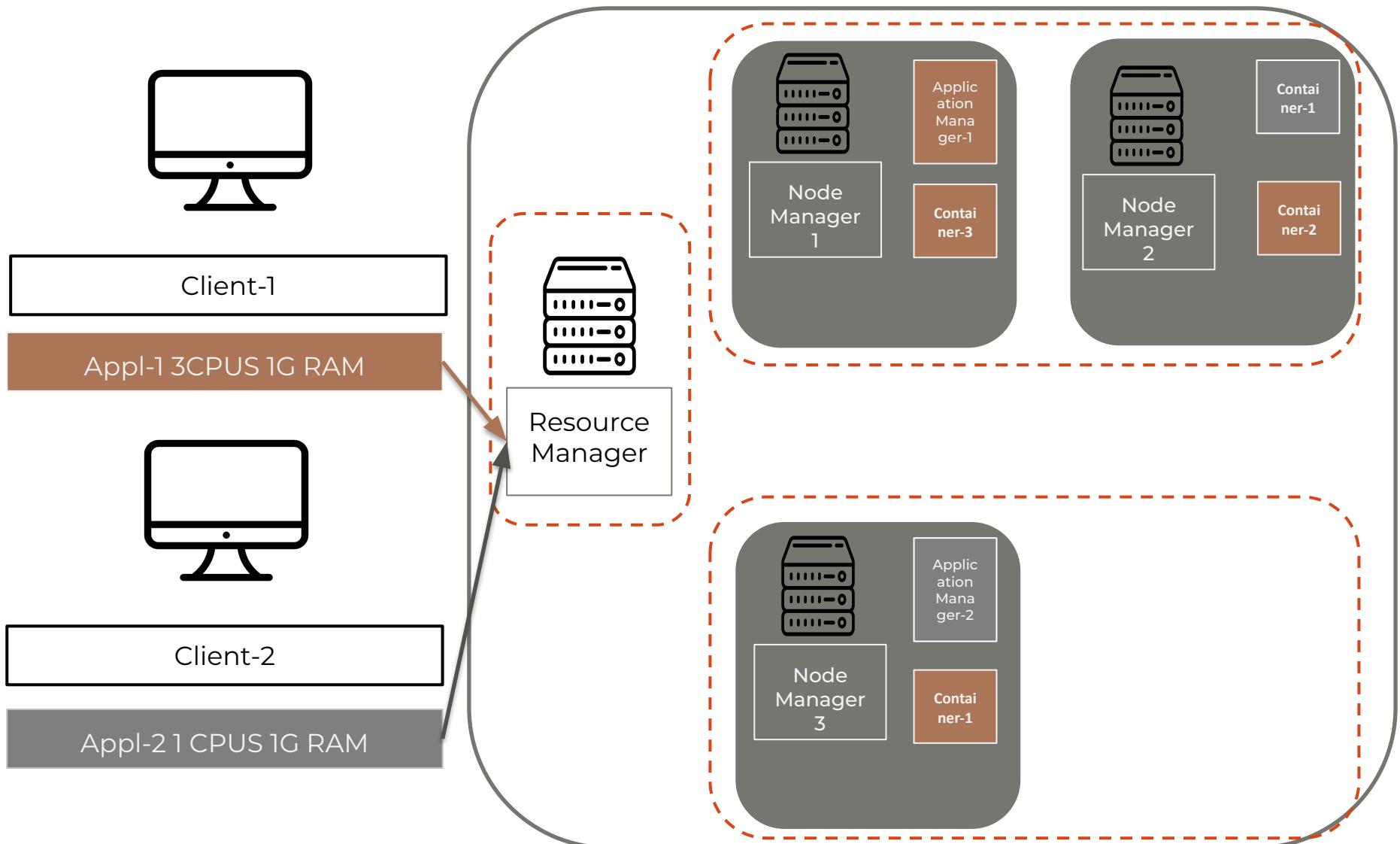
YARN - M&R Integracion



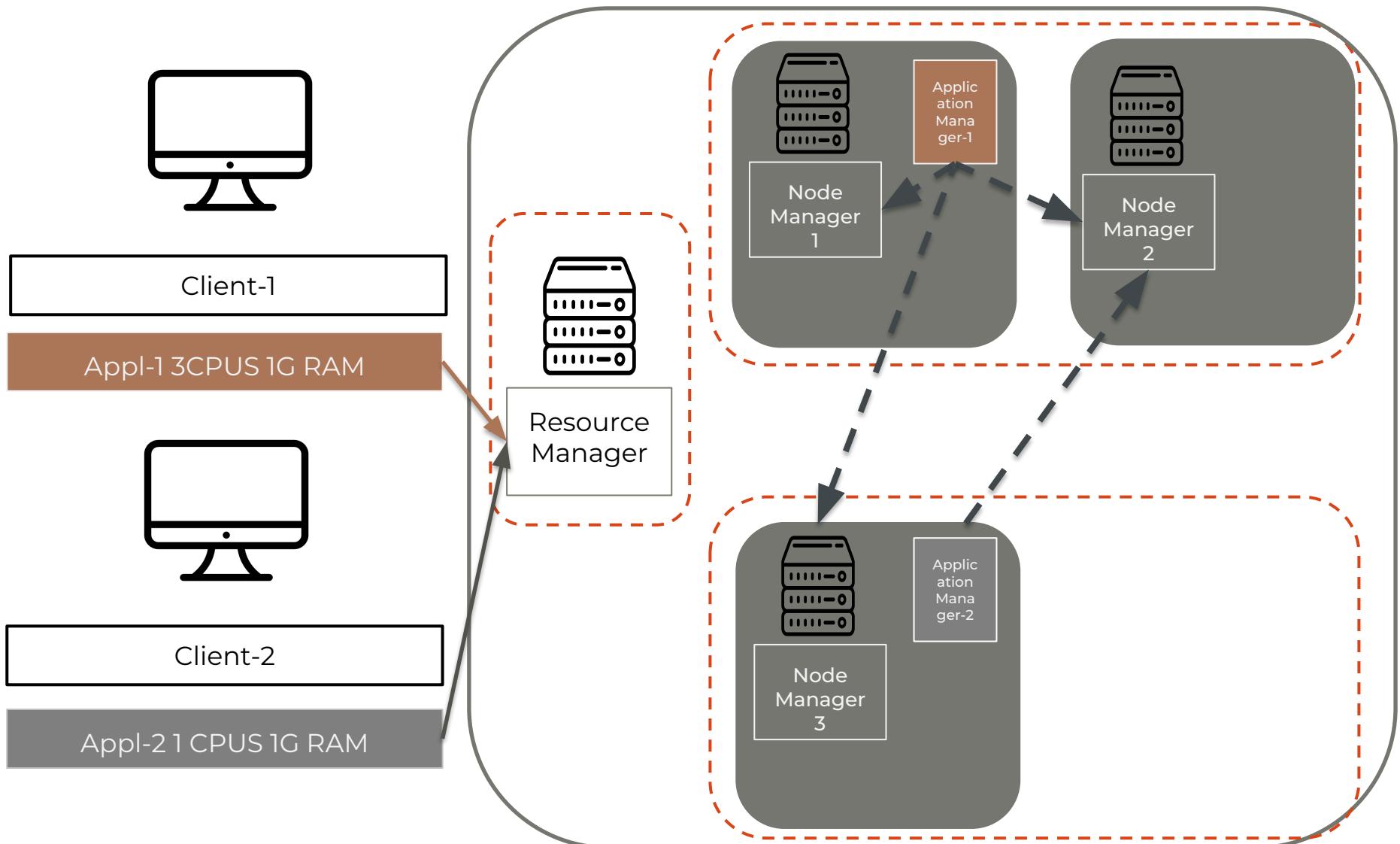
YARN - M&R Integracion



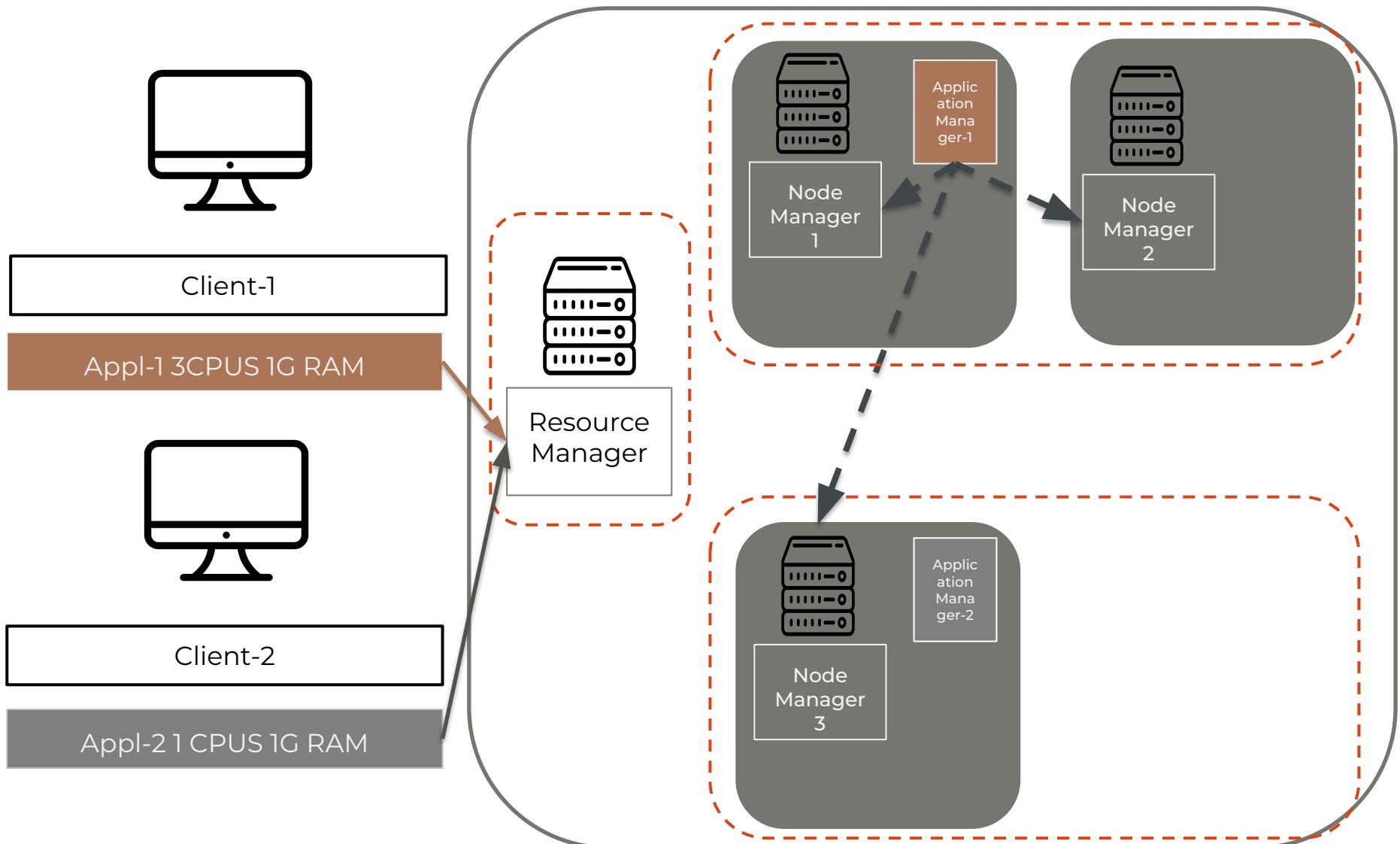
YARN - M&R Integracion



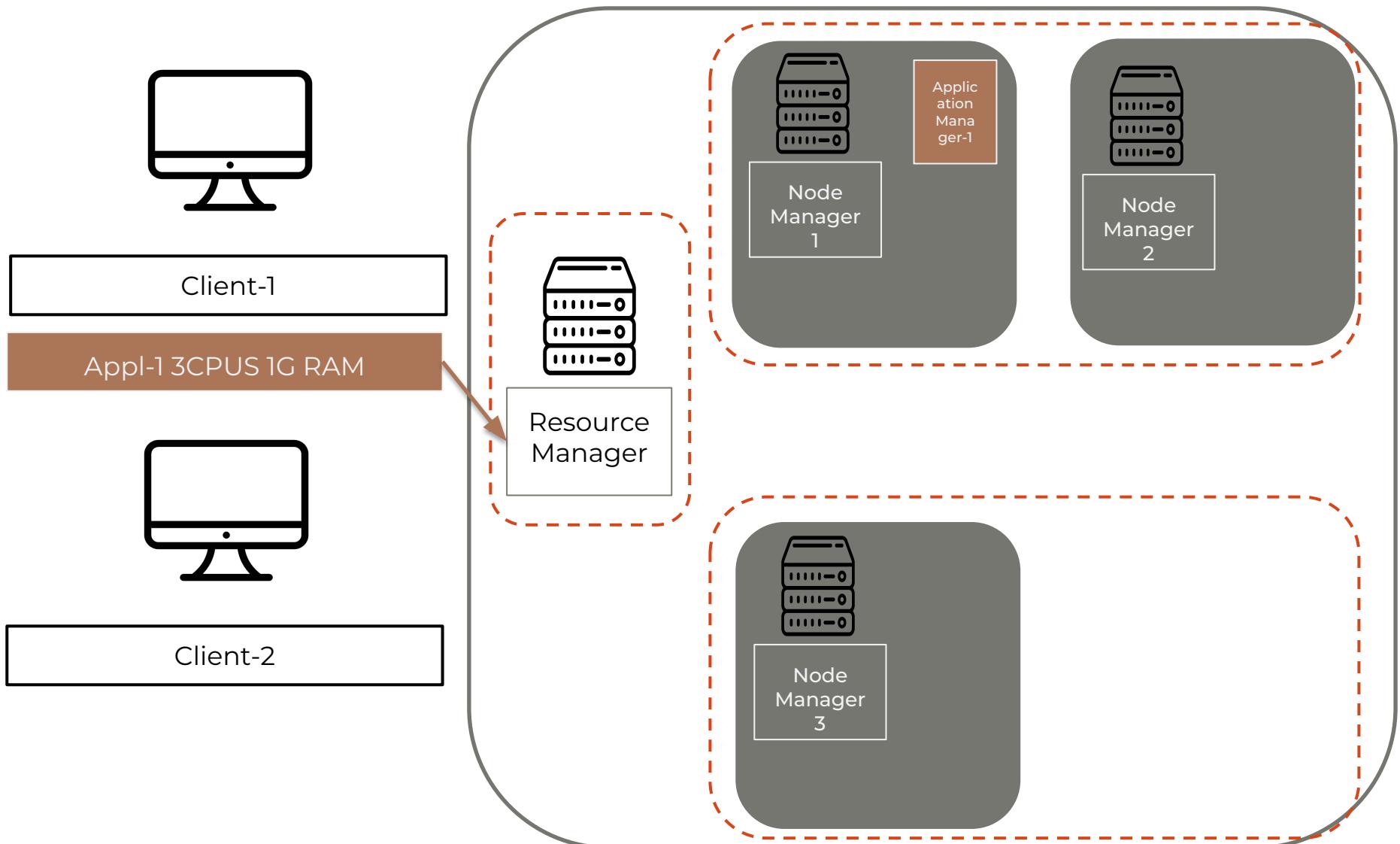
YARN - M&R Integracion



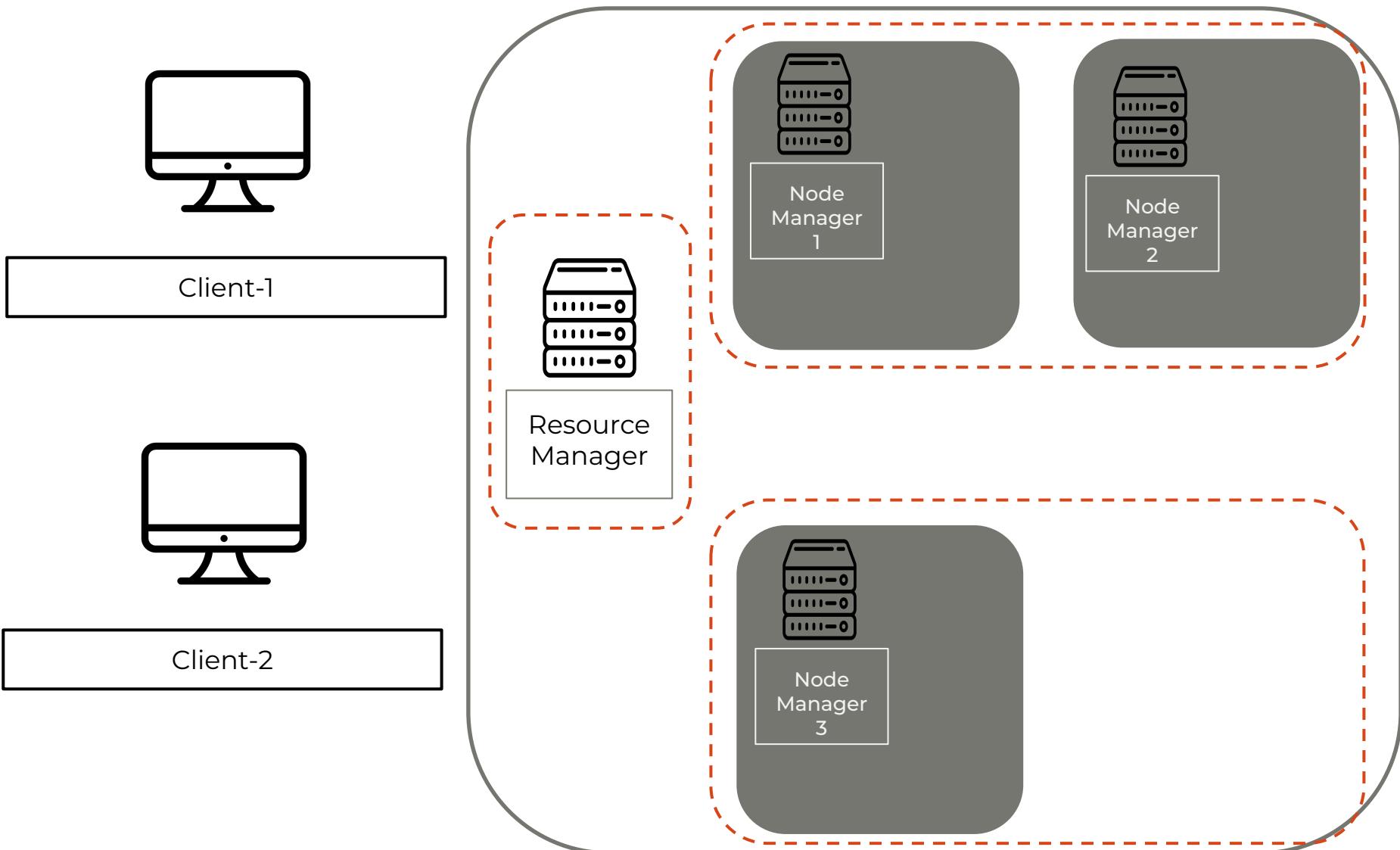
YARN - M&R Integracion



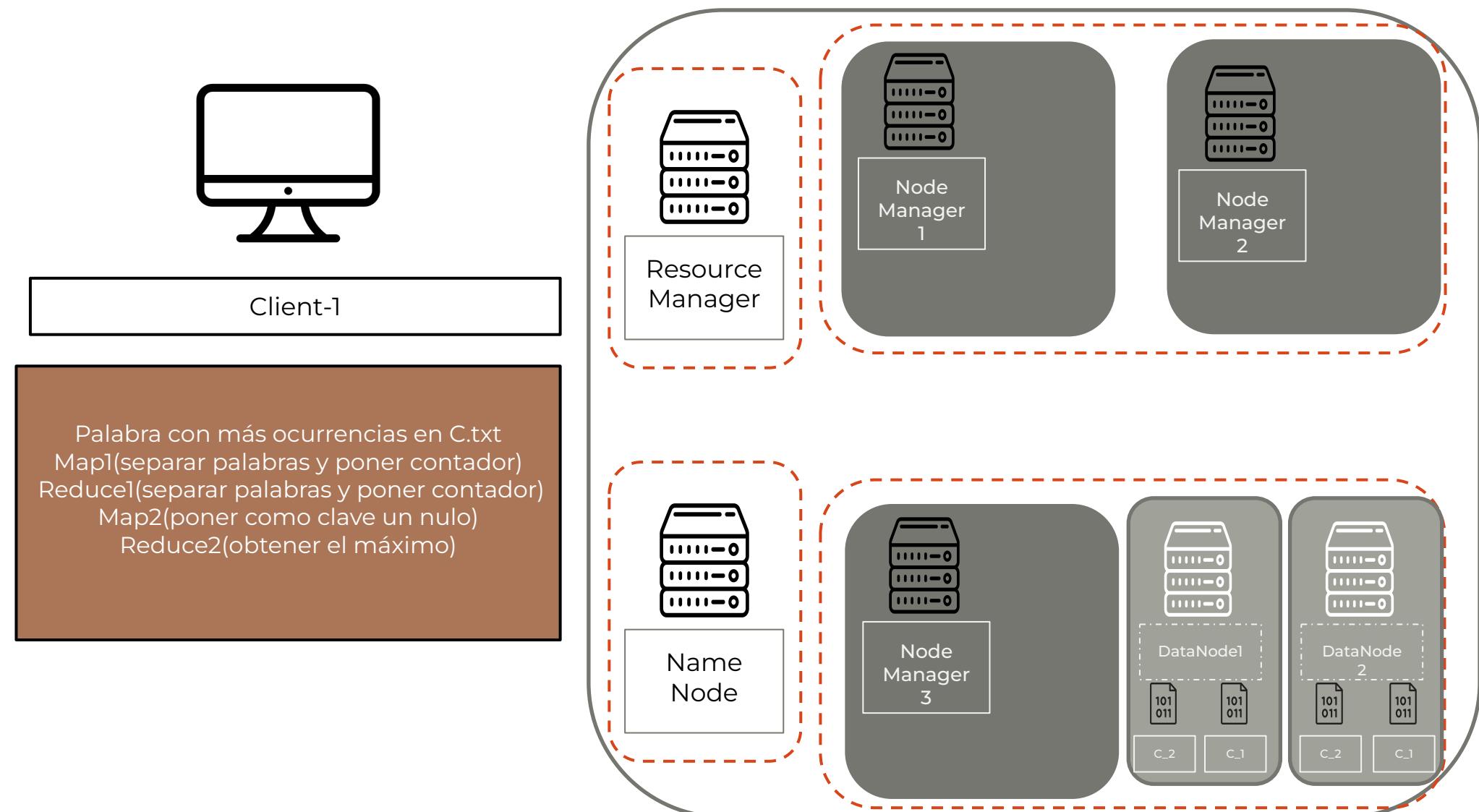
YARN - M&R Integracion



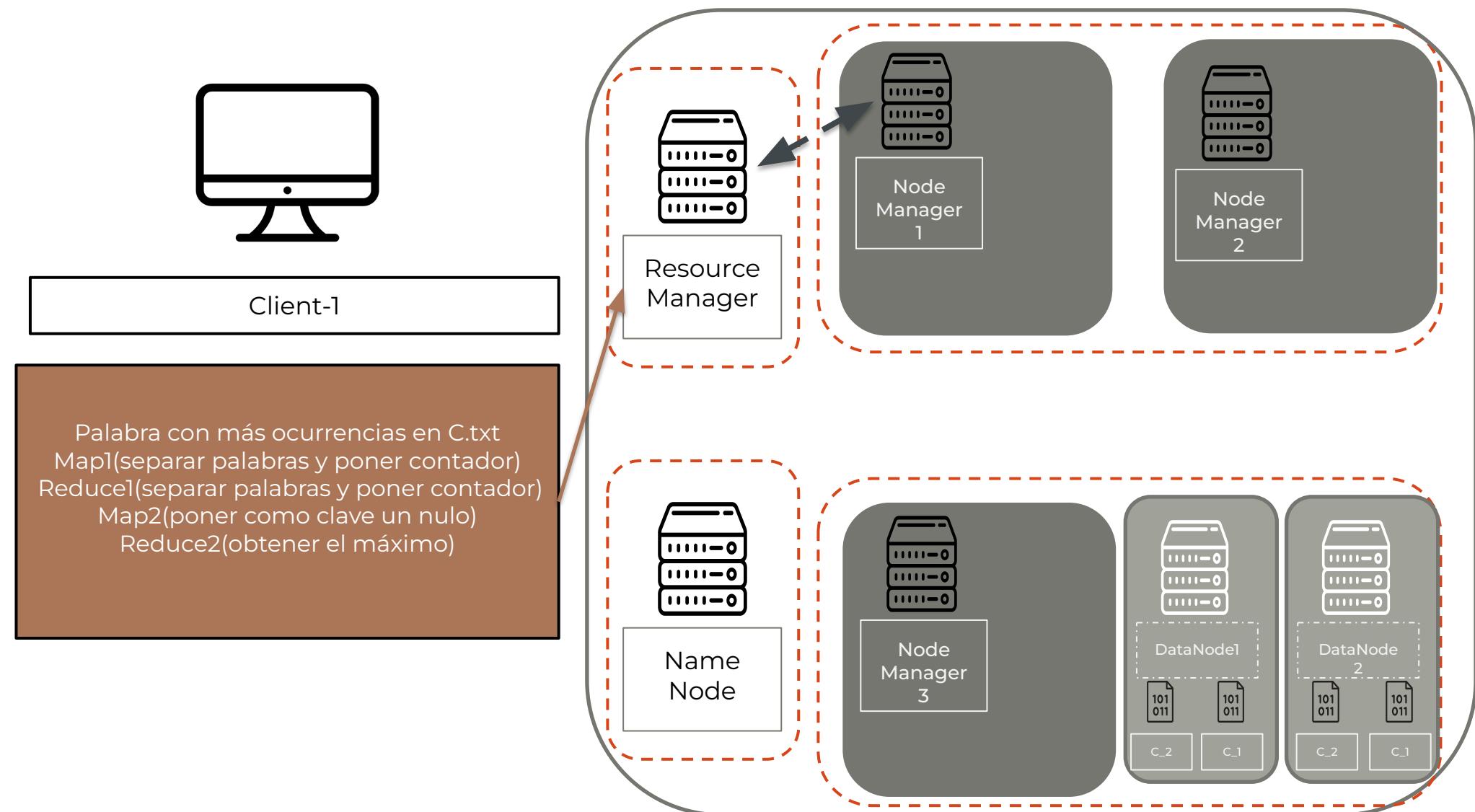
YARN - M&R Integracion



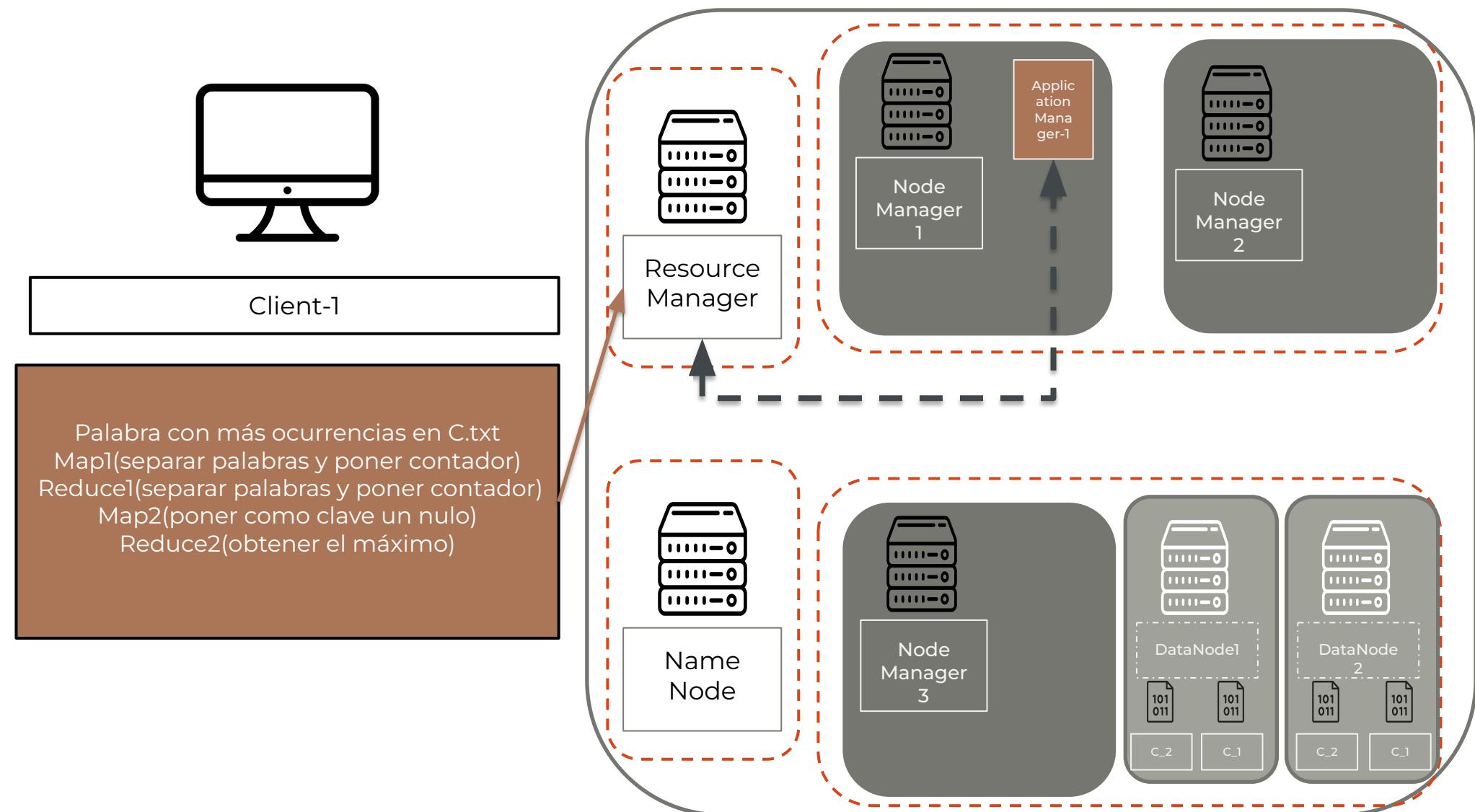
YARN - M&R - HDFS Integración



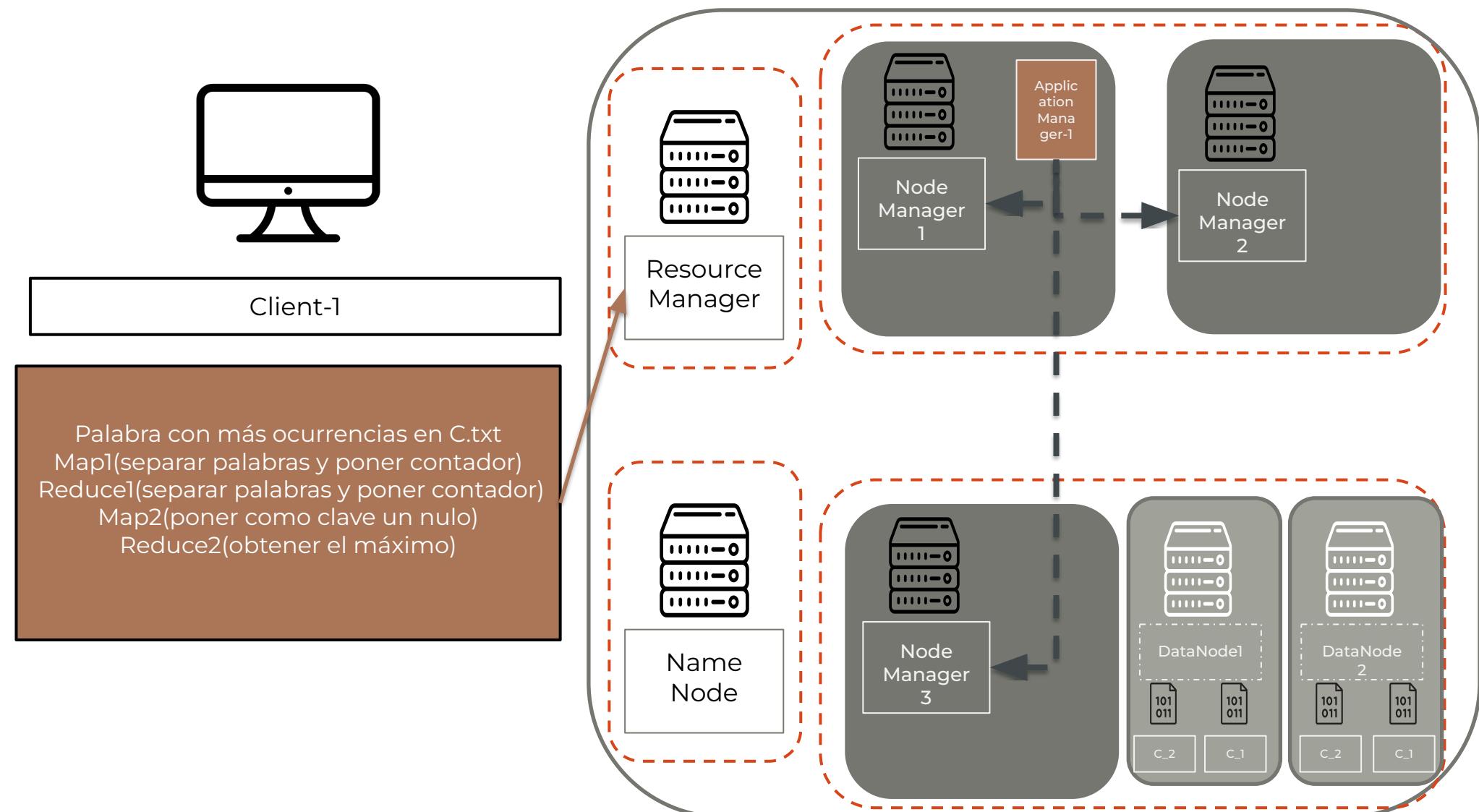
YARN - M&R - HDFS Integración



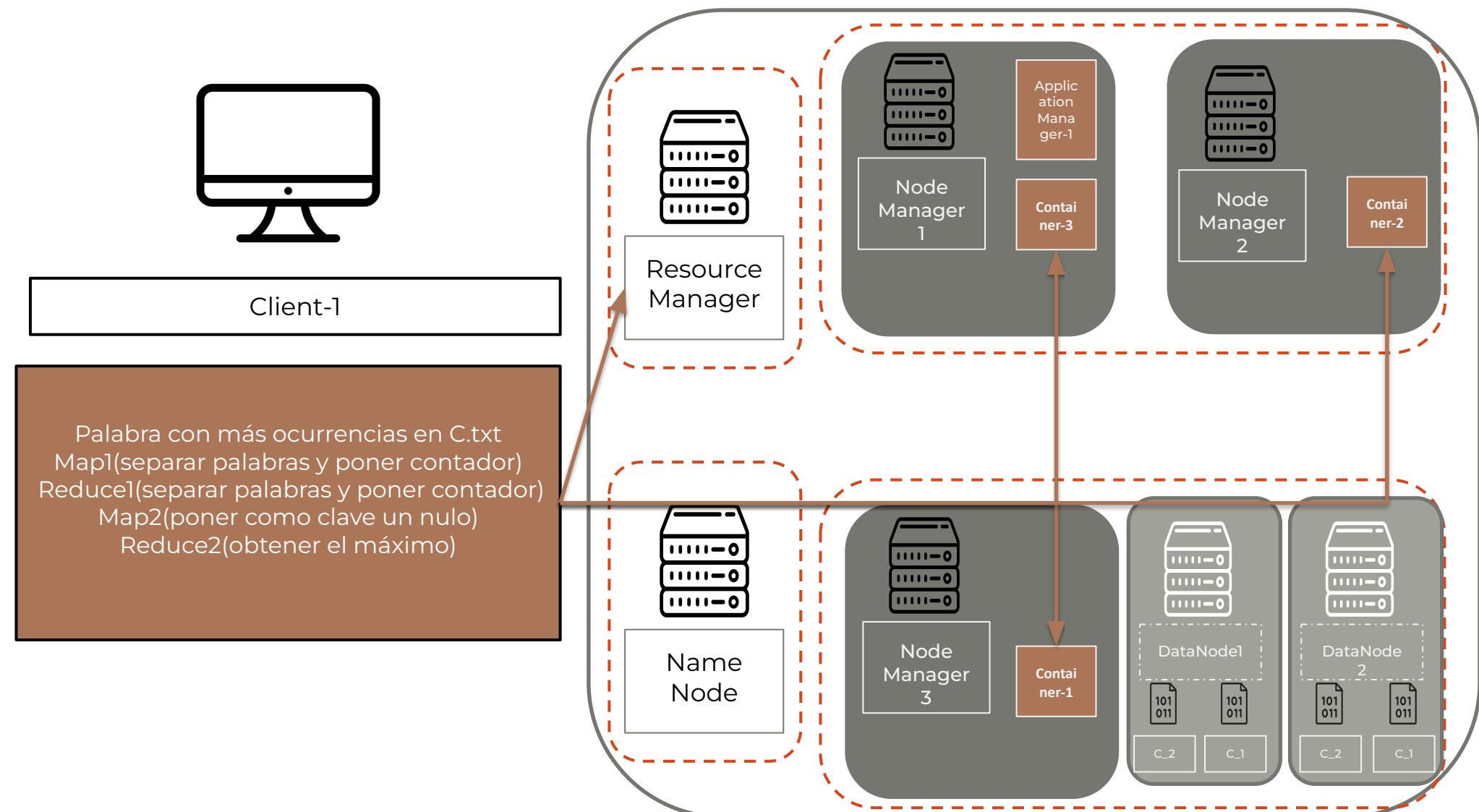
YARN - M&R - HDFS Integración



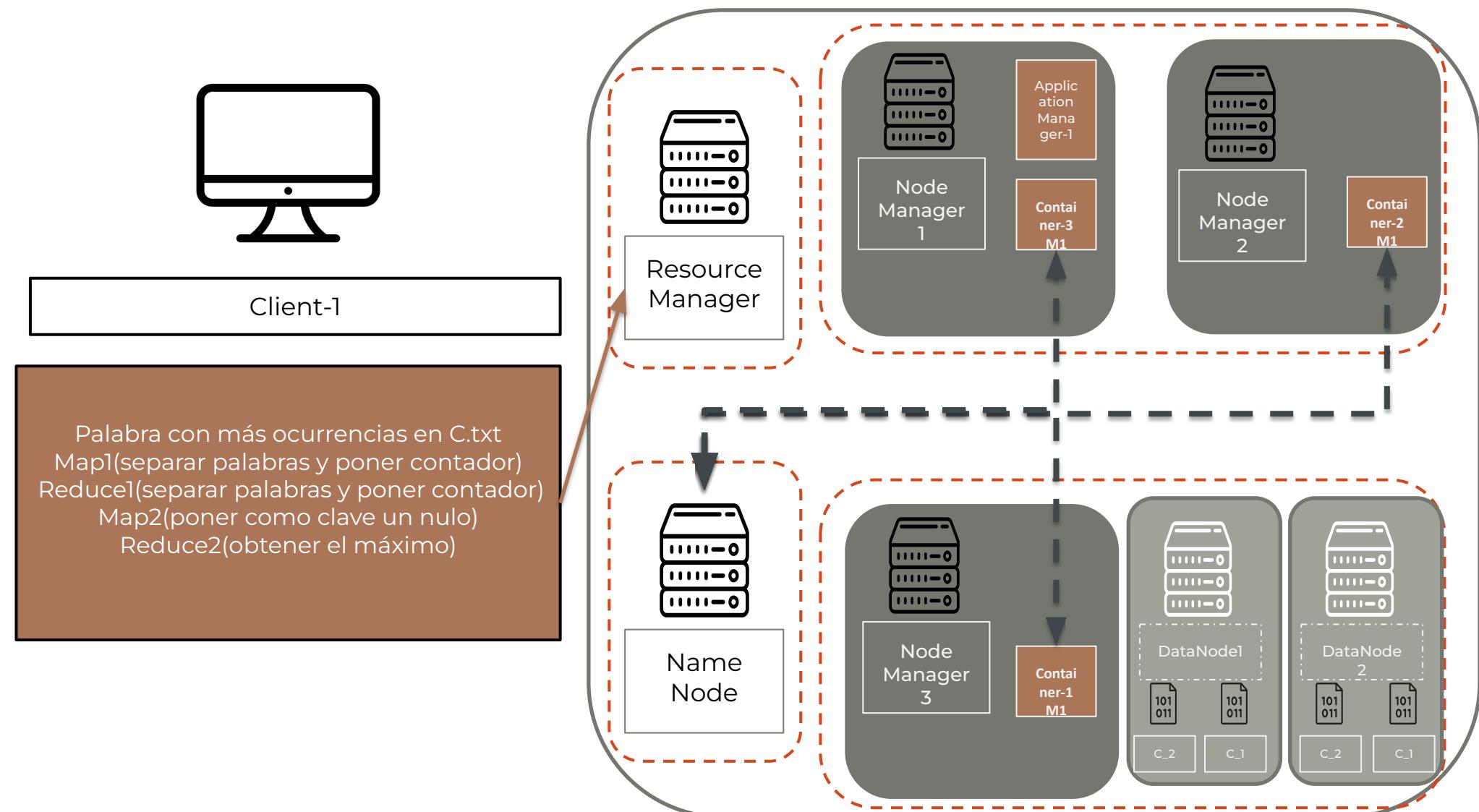
YARN - M&R - HDFS Integración



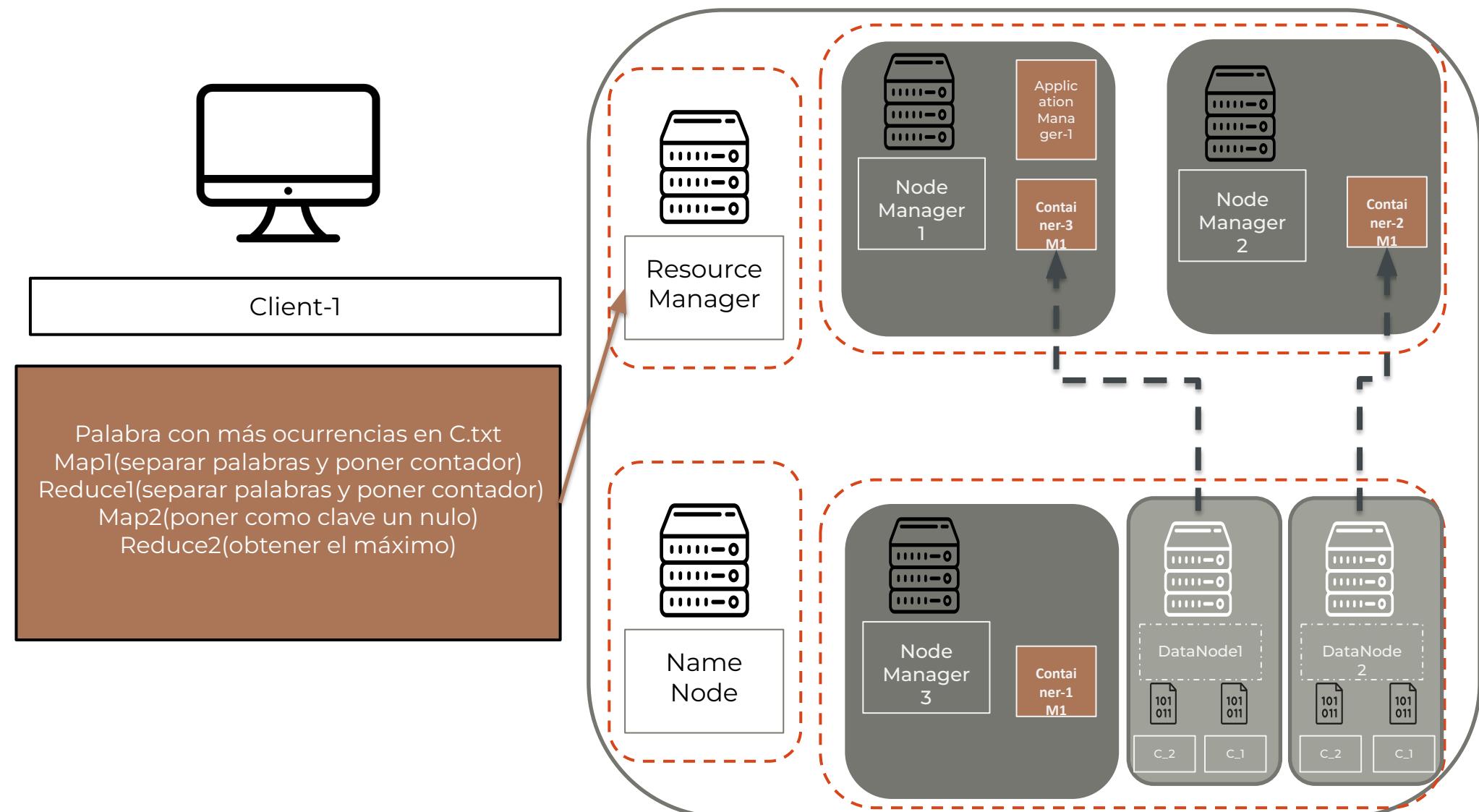
YARN - M&R - HDFS Integración



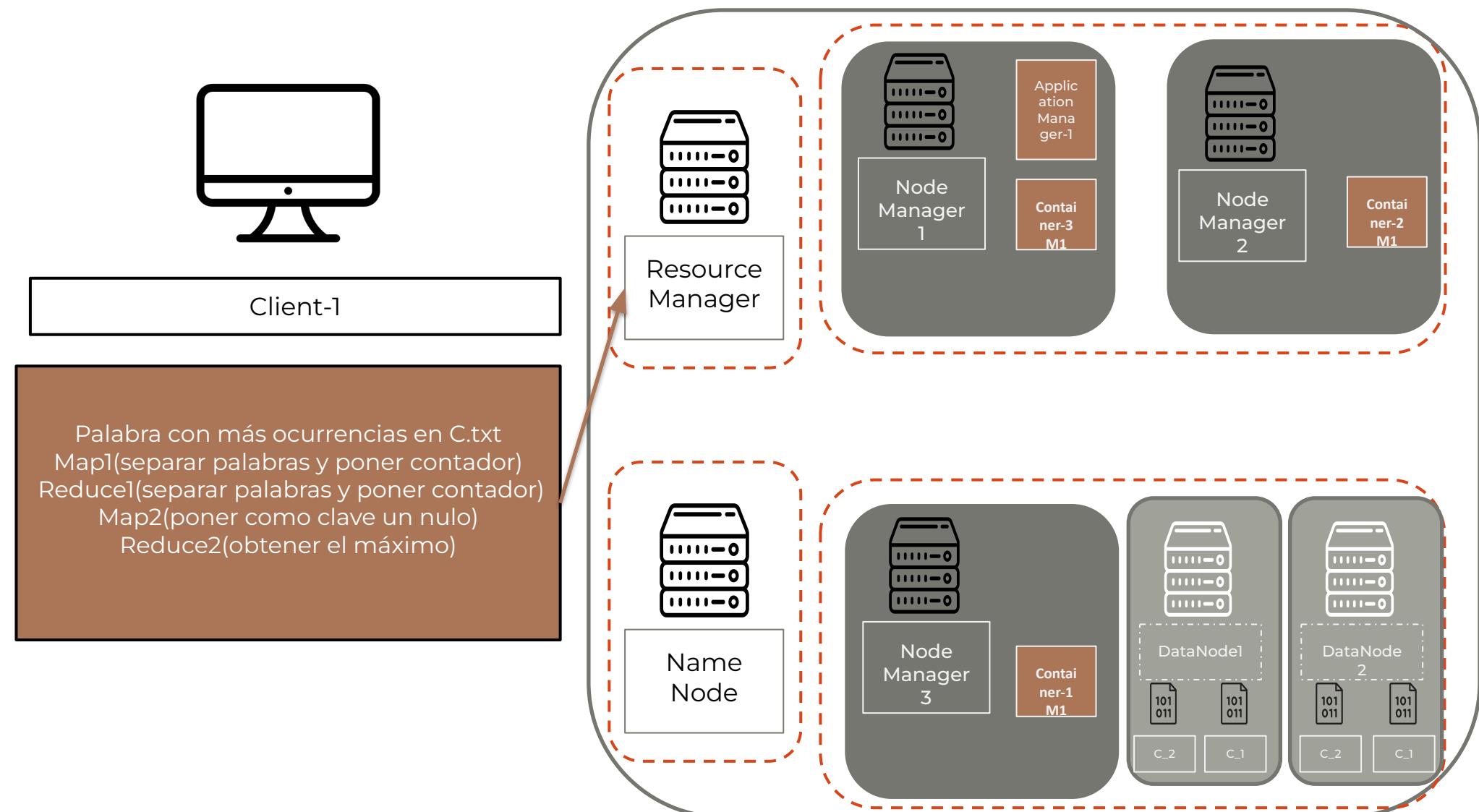
YARN - M&R - HDFS Integración



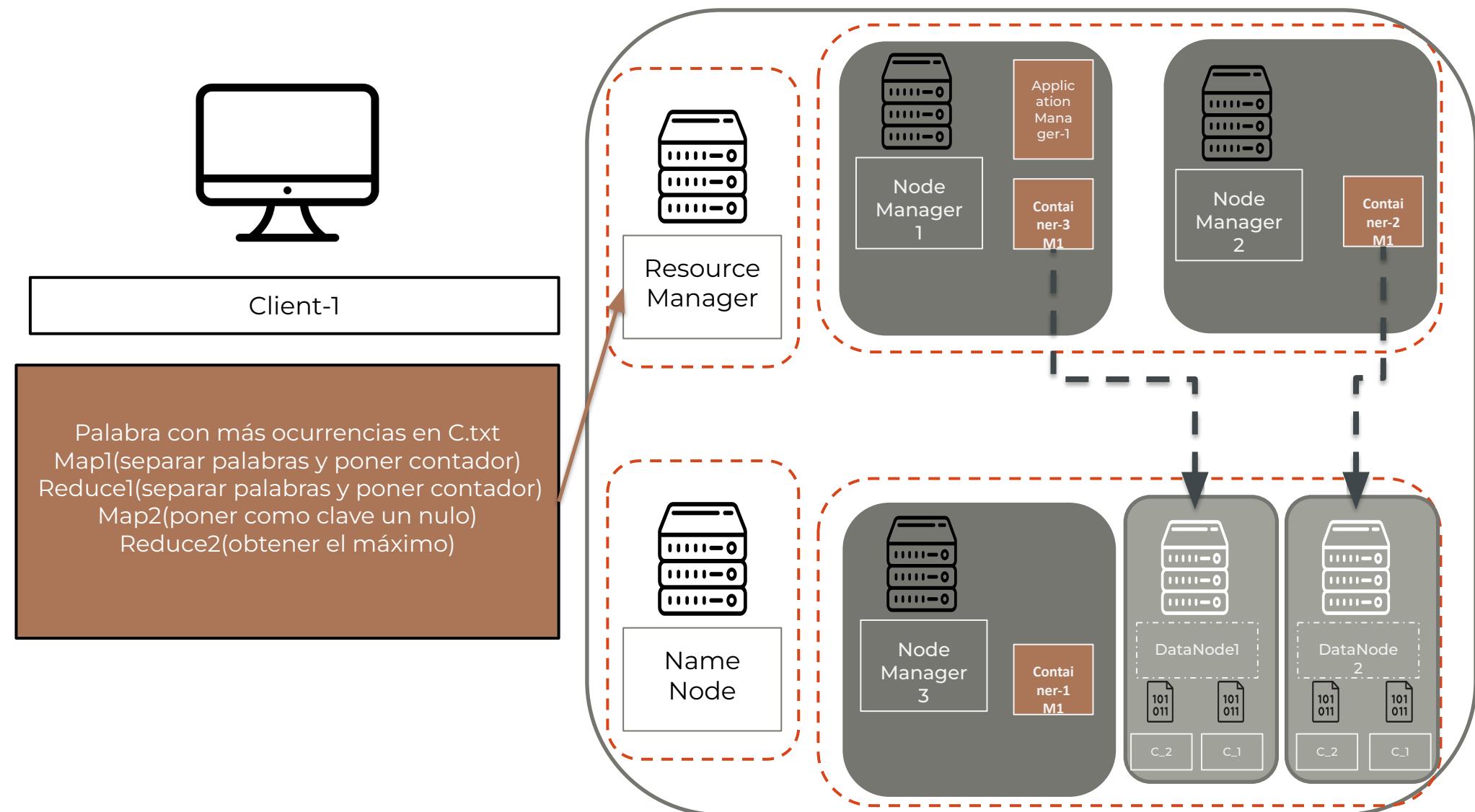
YARN - M&R - HDFS Integración



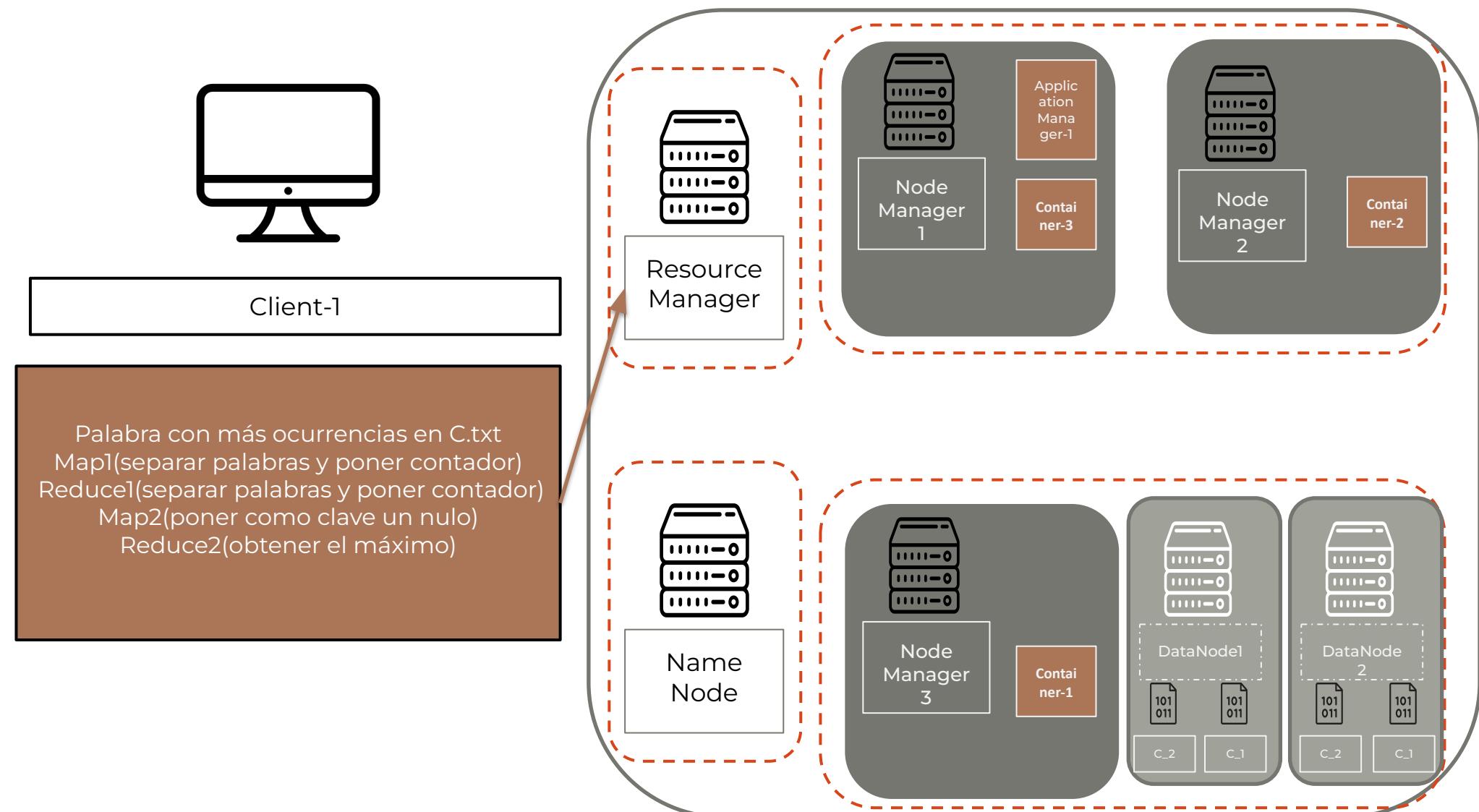
YARN - M&R - HDFS Integración



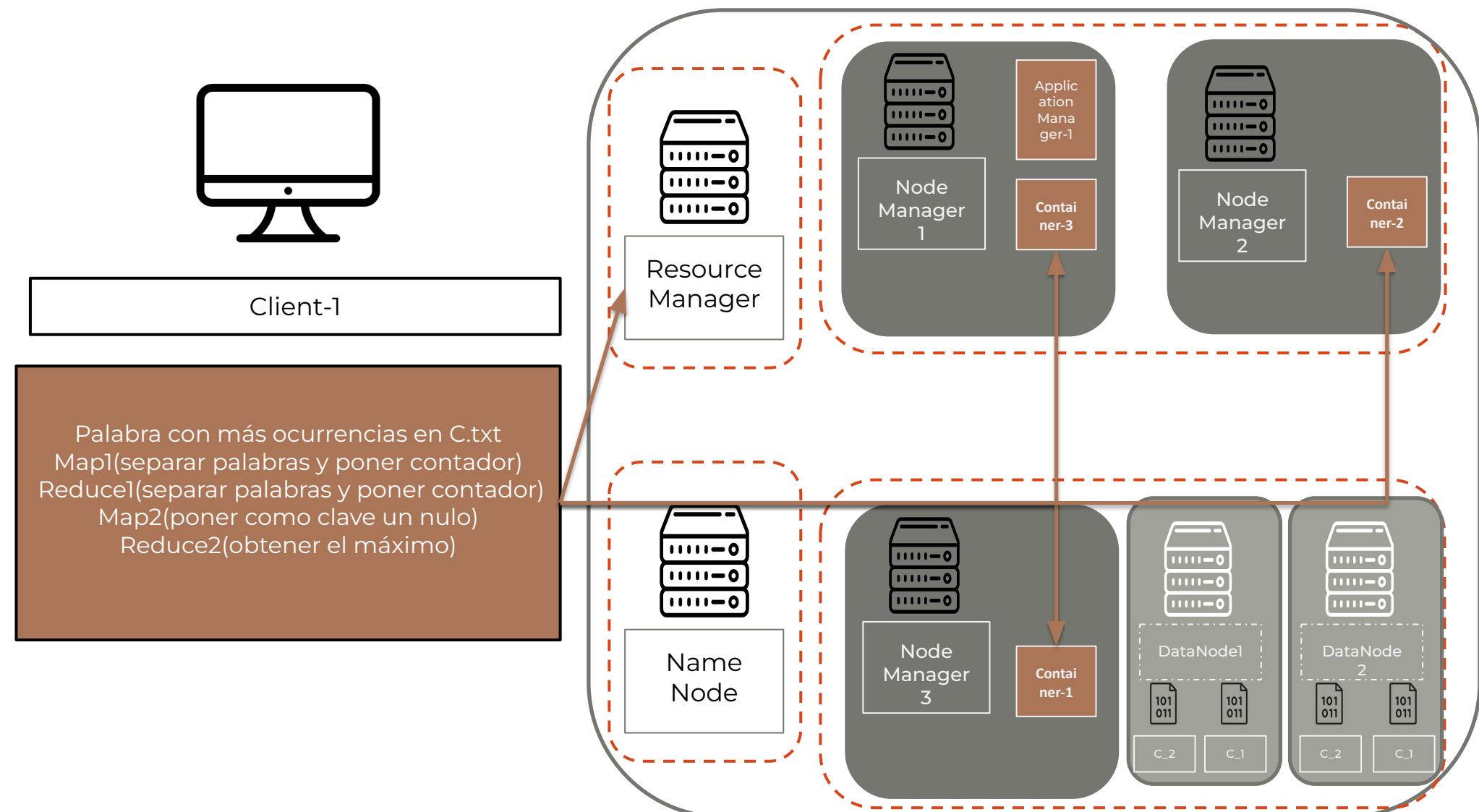
YARN - M&R - HDFS Integración



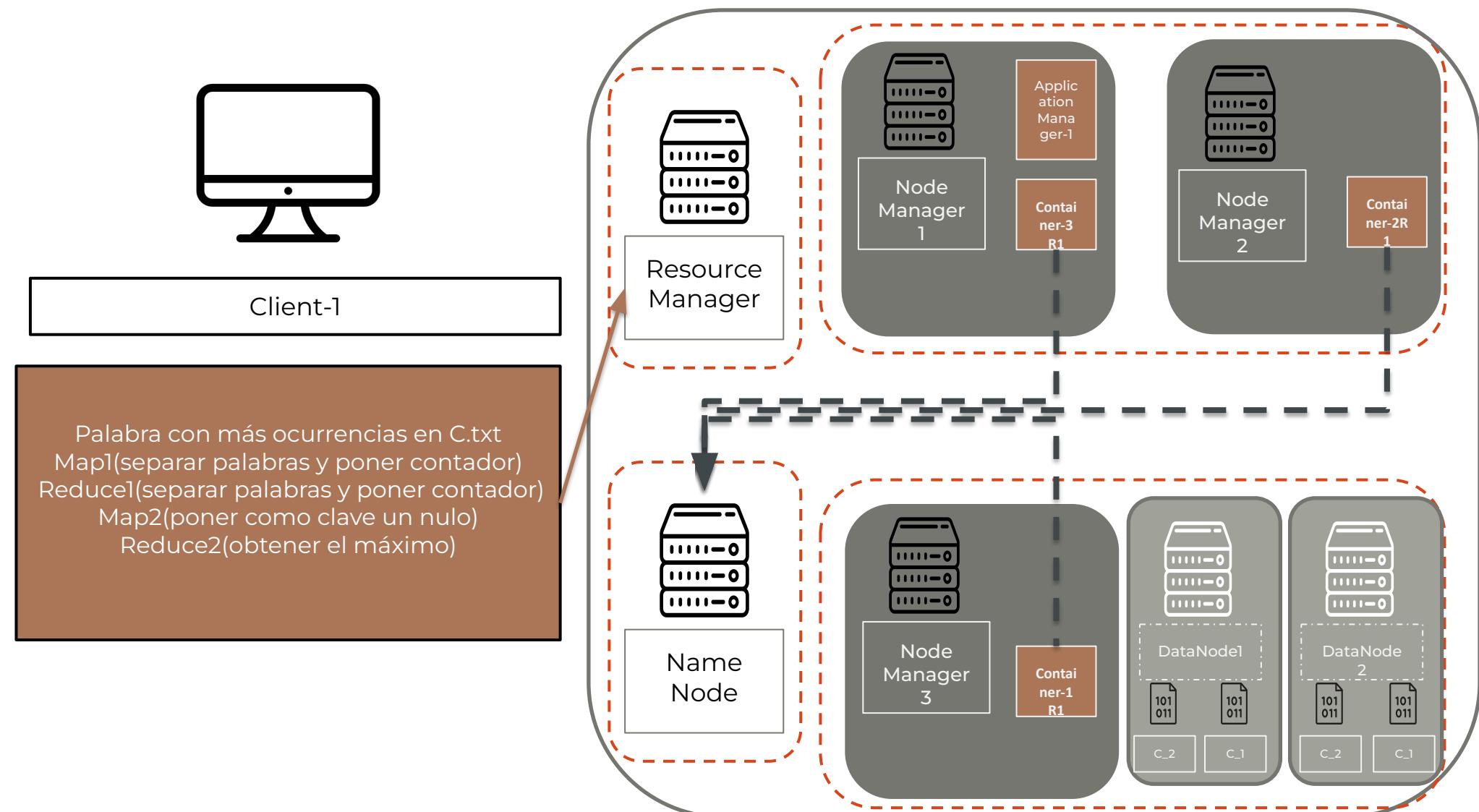
YARN - M&R - HDFS Integración



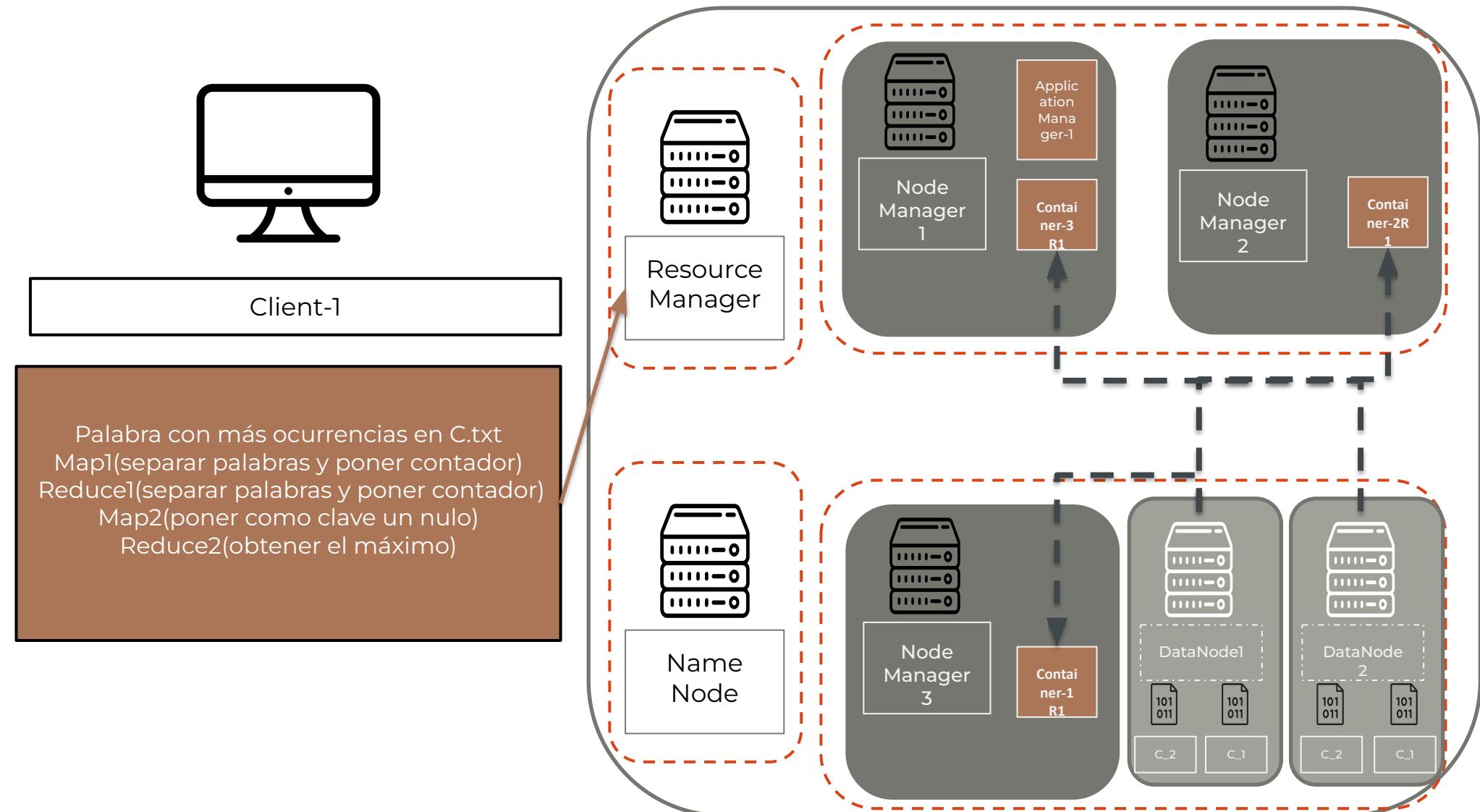
YARN - M&R - HDFS Integración



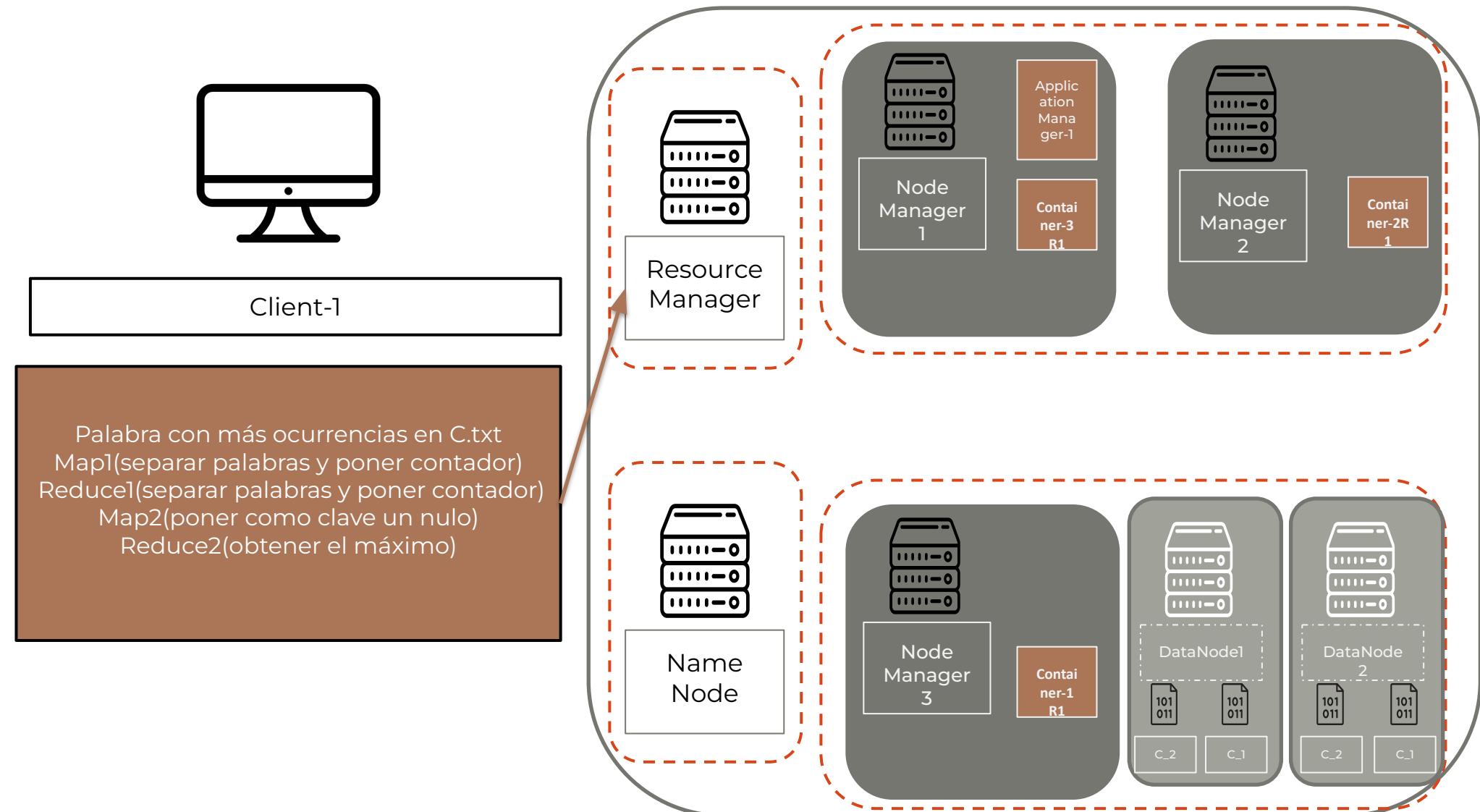
YARN - M&R - HDFS Integración



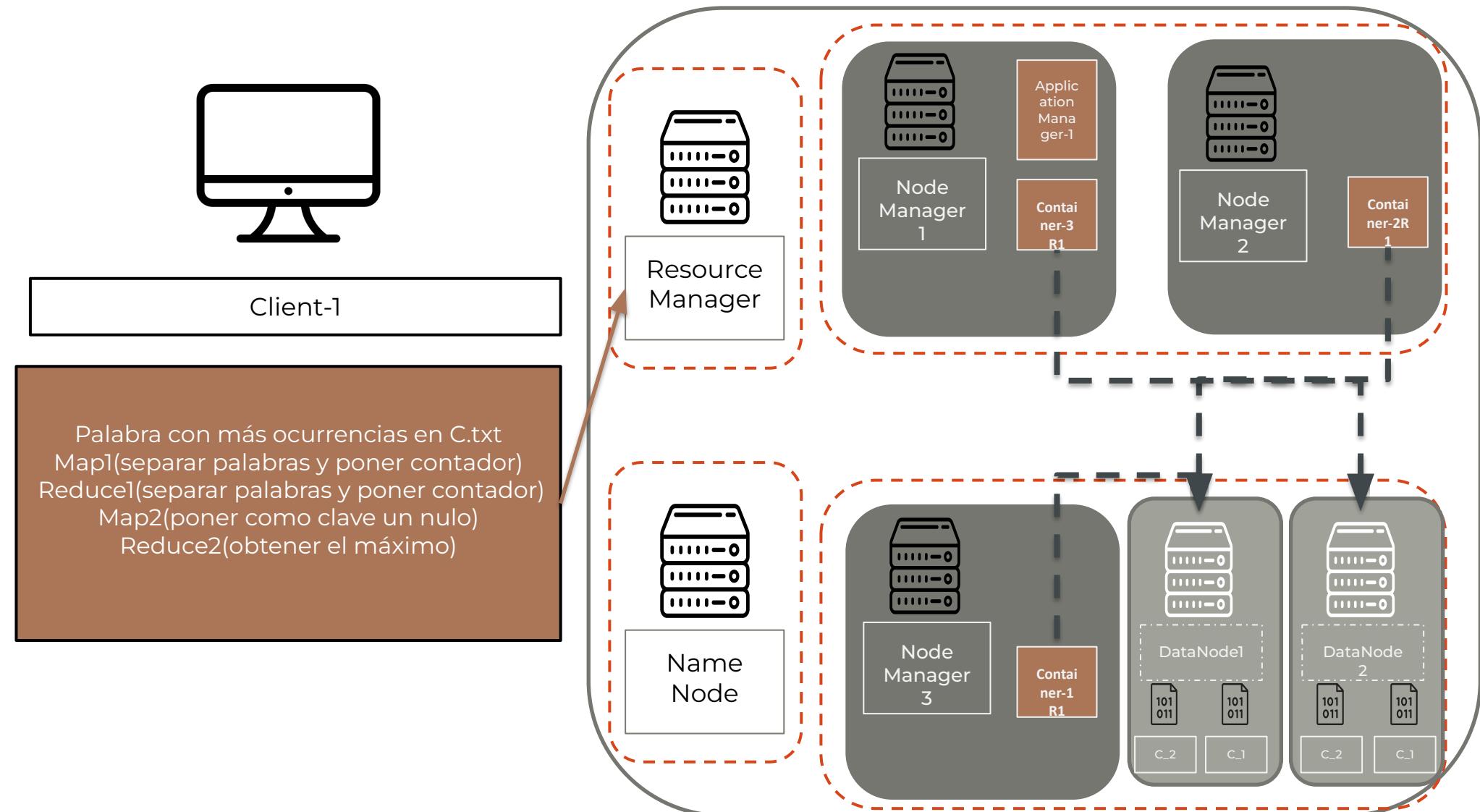
YARN - M&R - HDFS Integración



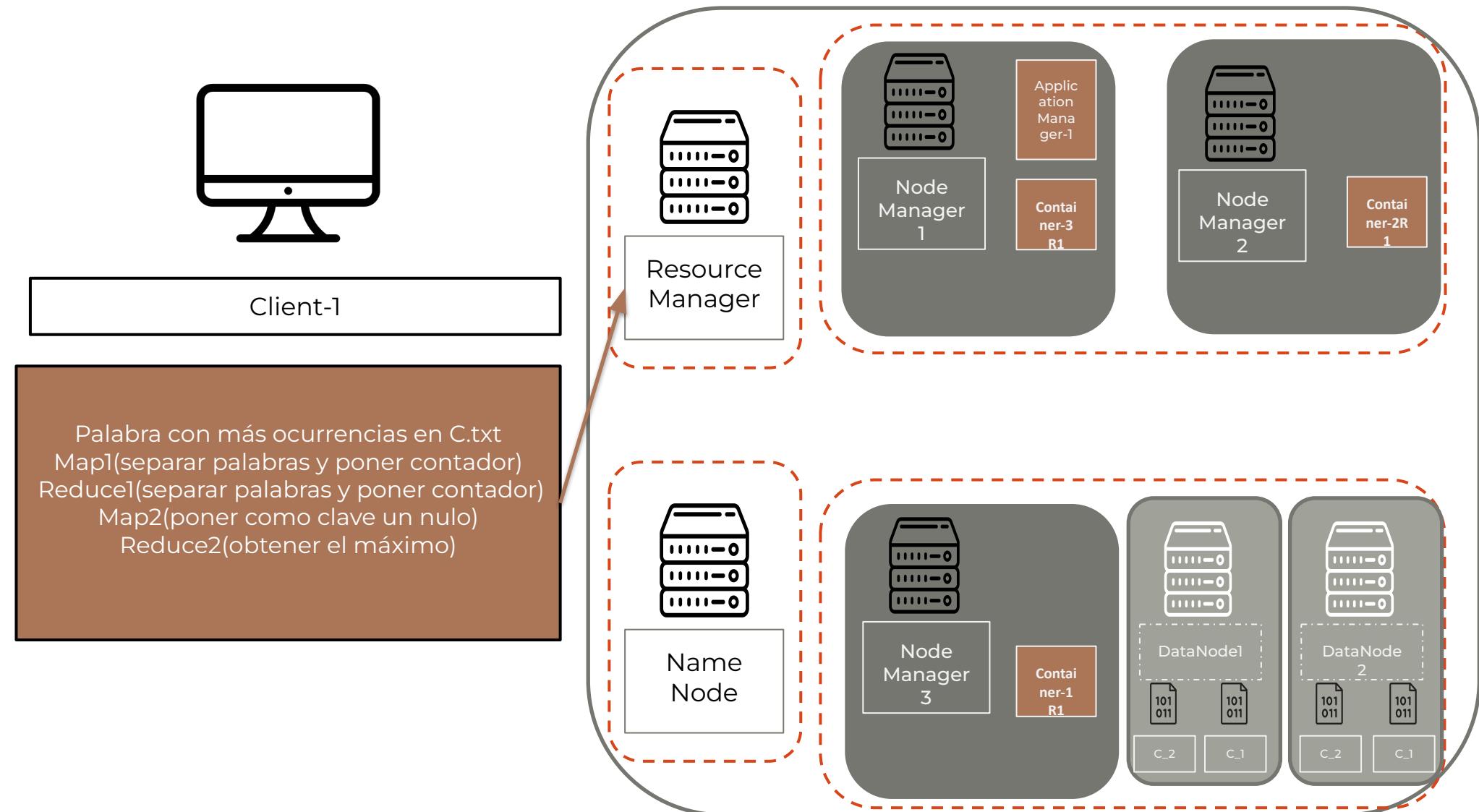
YARN - M&R - HDFS Integración



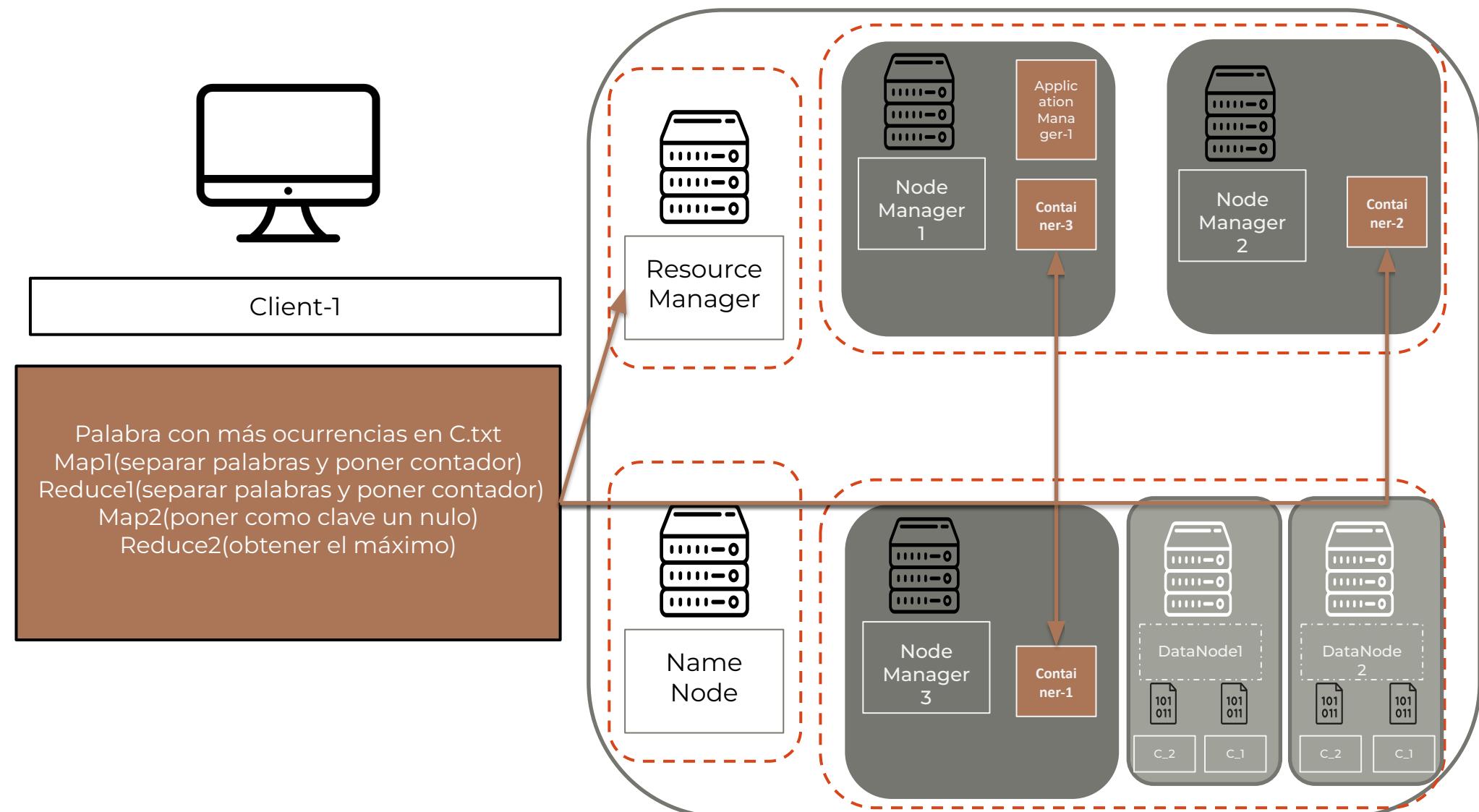
YARN - M&R - HDFS Integración



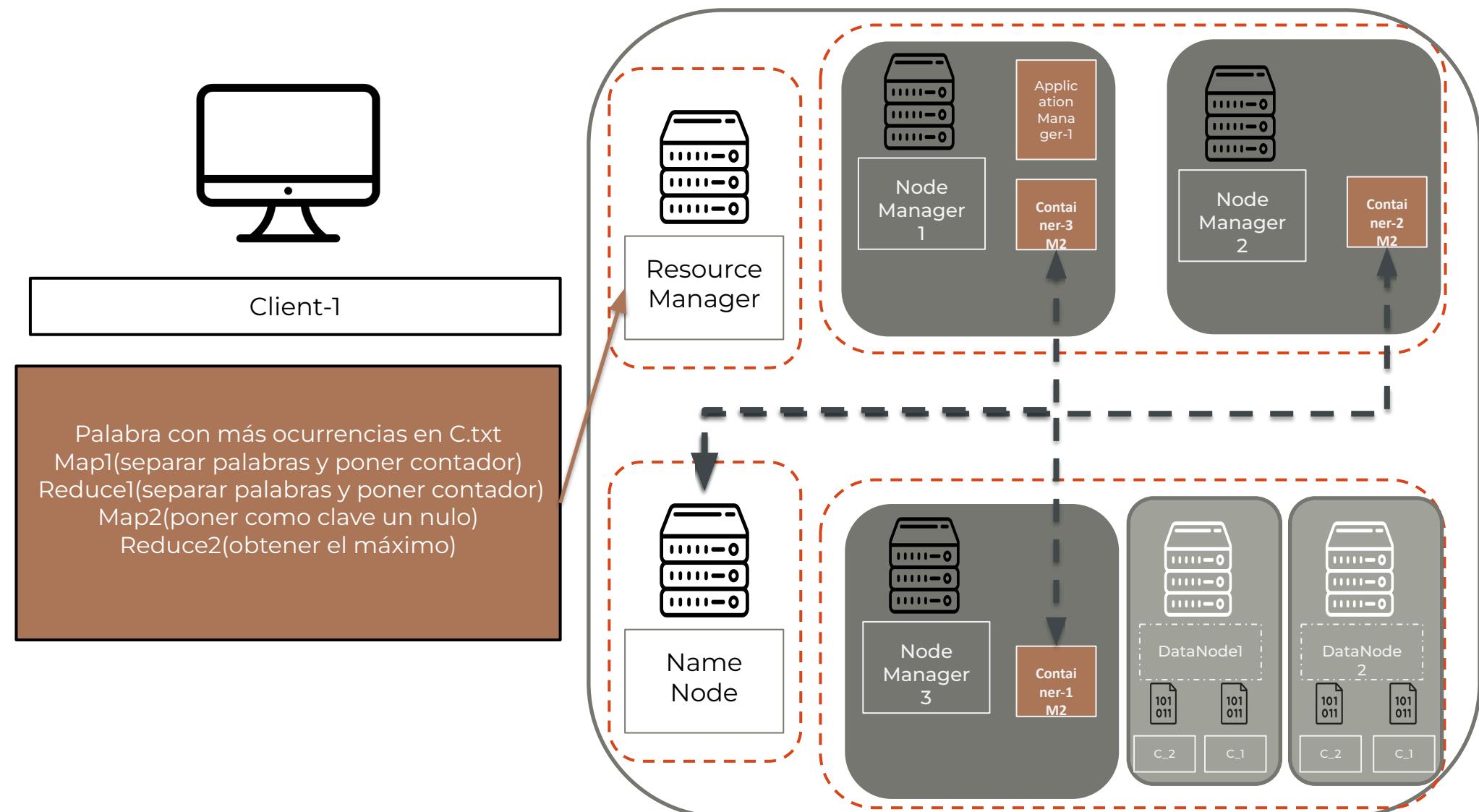
YARN - M&R - HDFS Integración



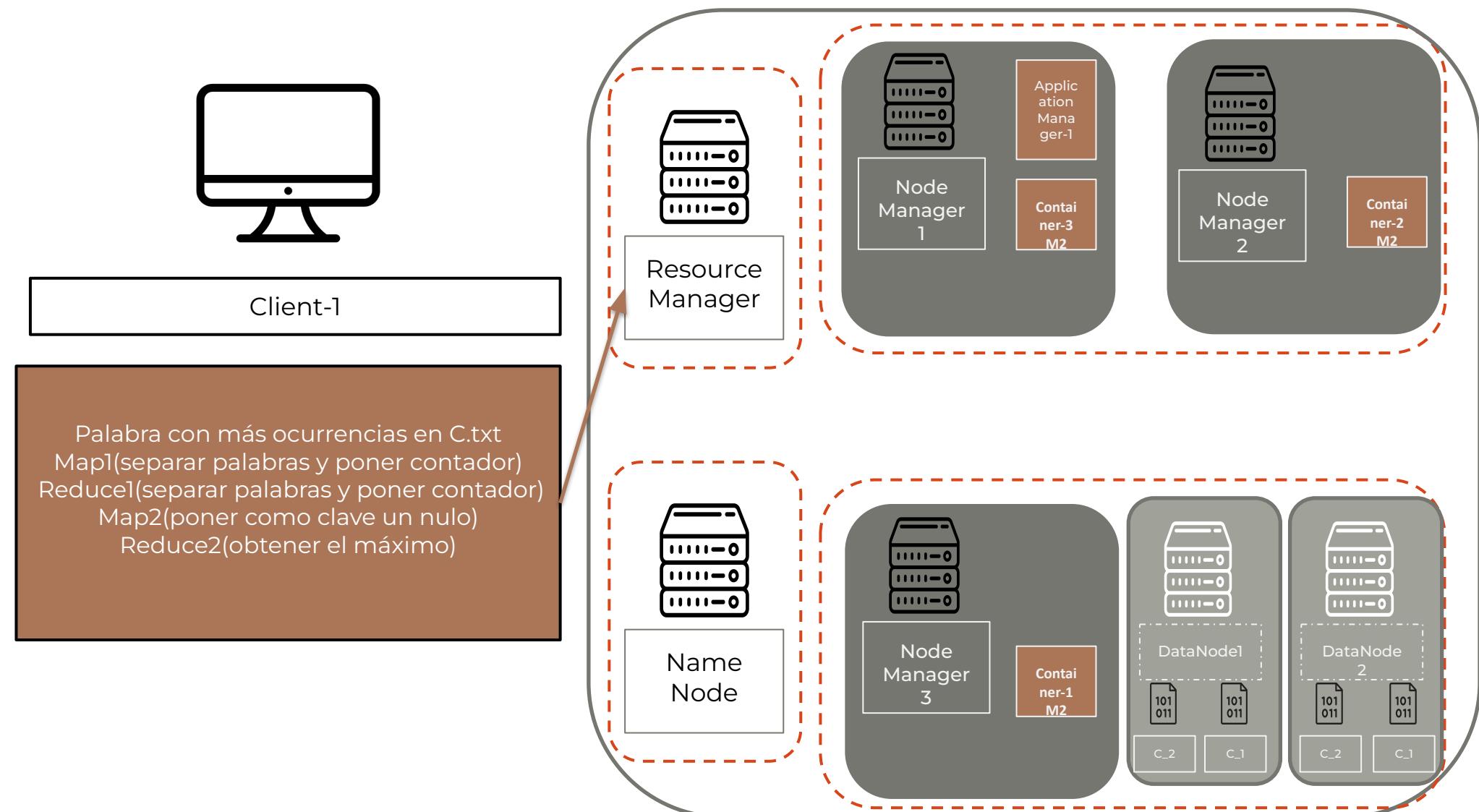
YARN - M&R - HDFS Integración



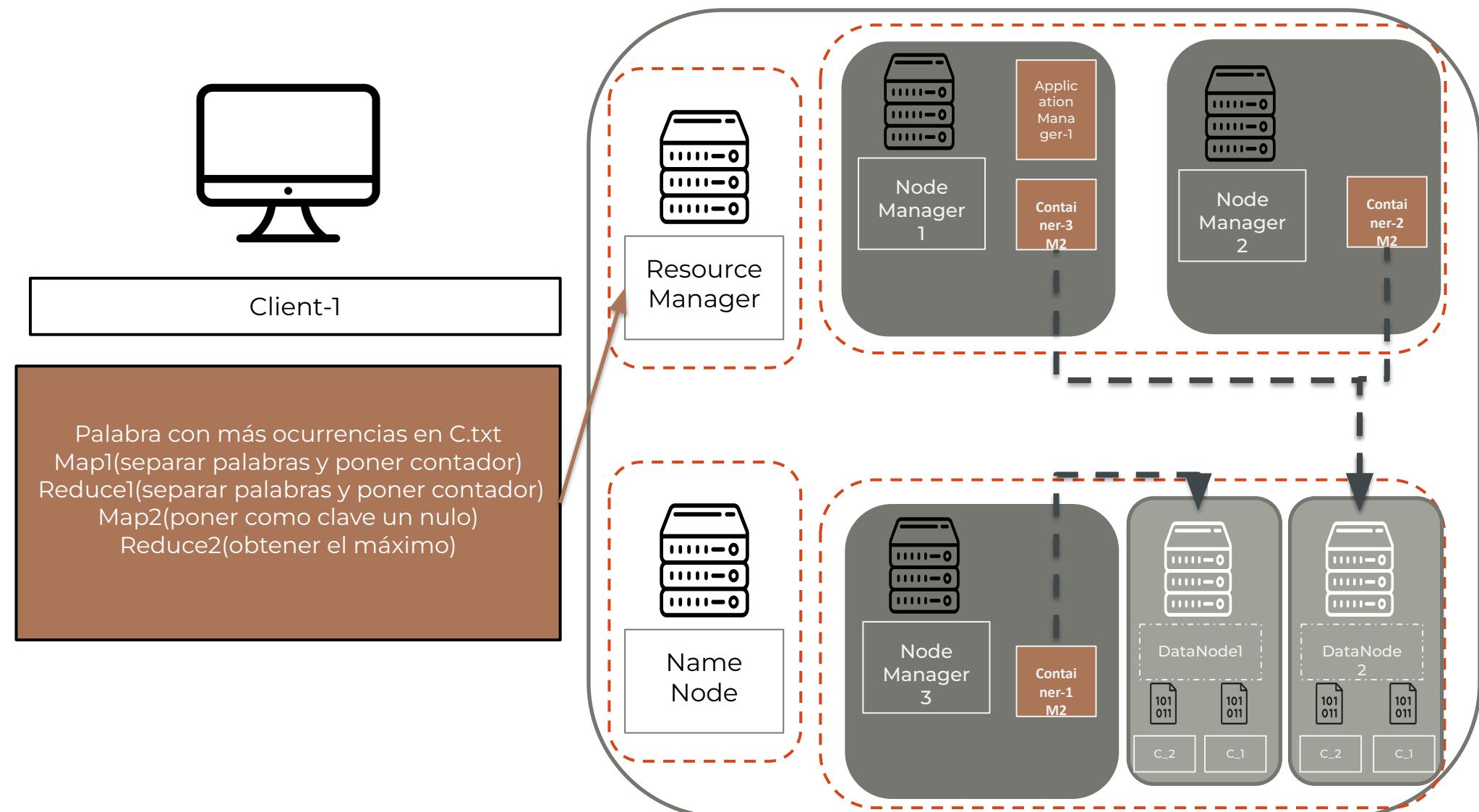
YARN - M&R - HDFS Integración



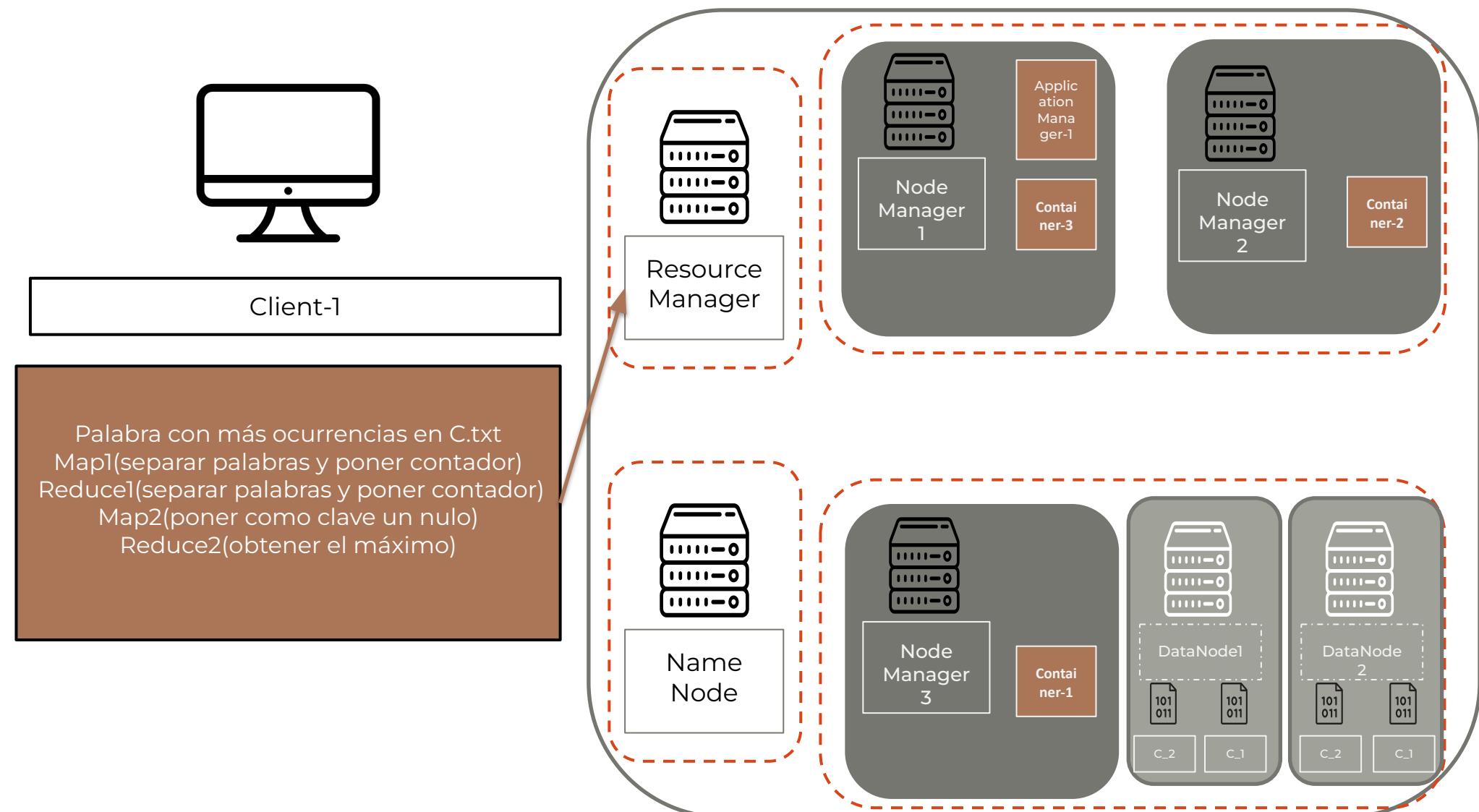
YARN - M&R - HDFS Integración



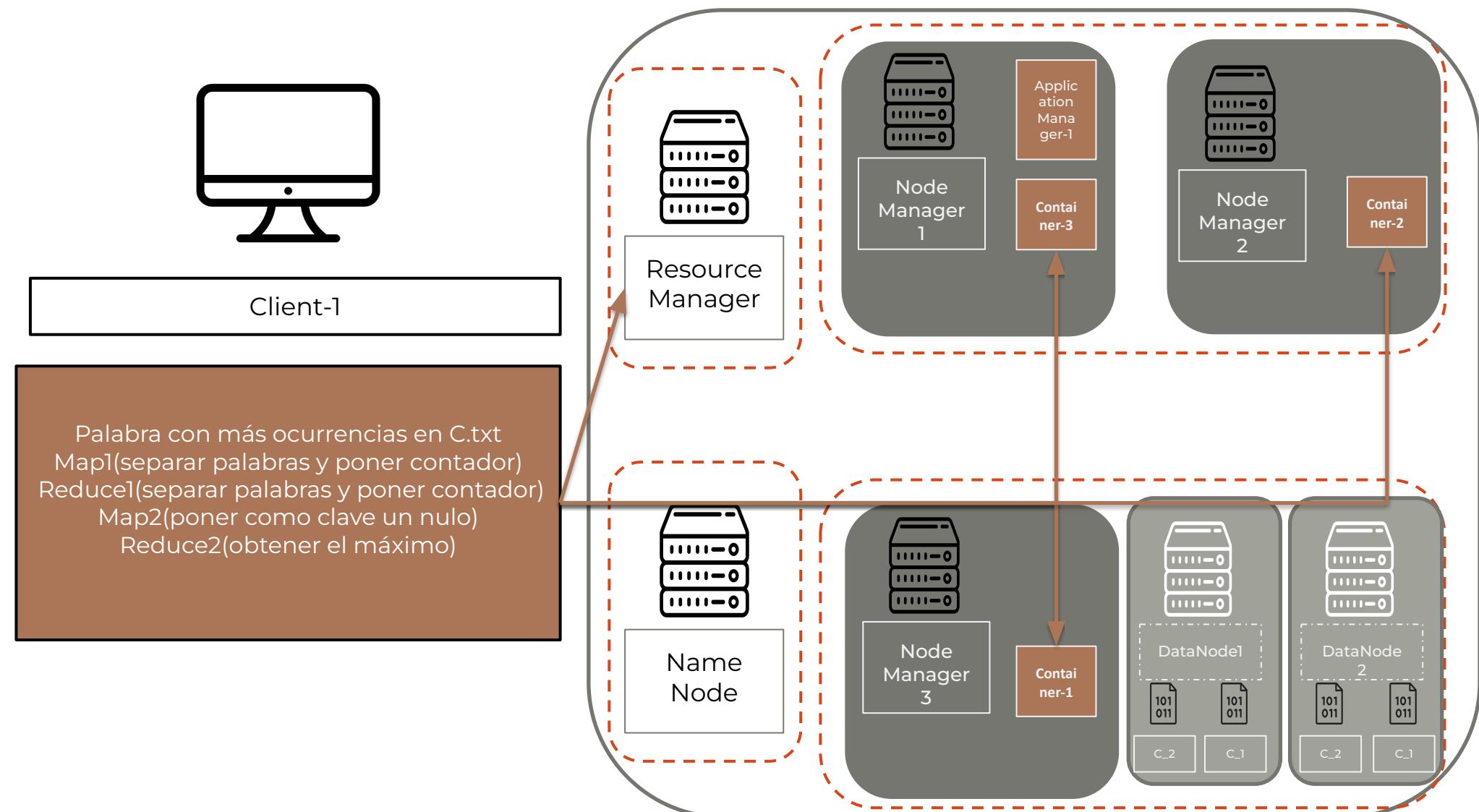
YARN - M&R - HDFS Integración



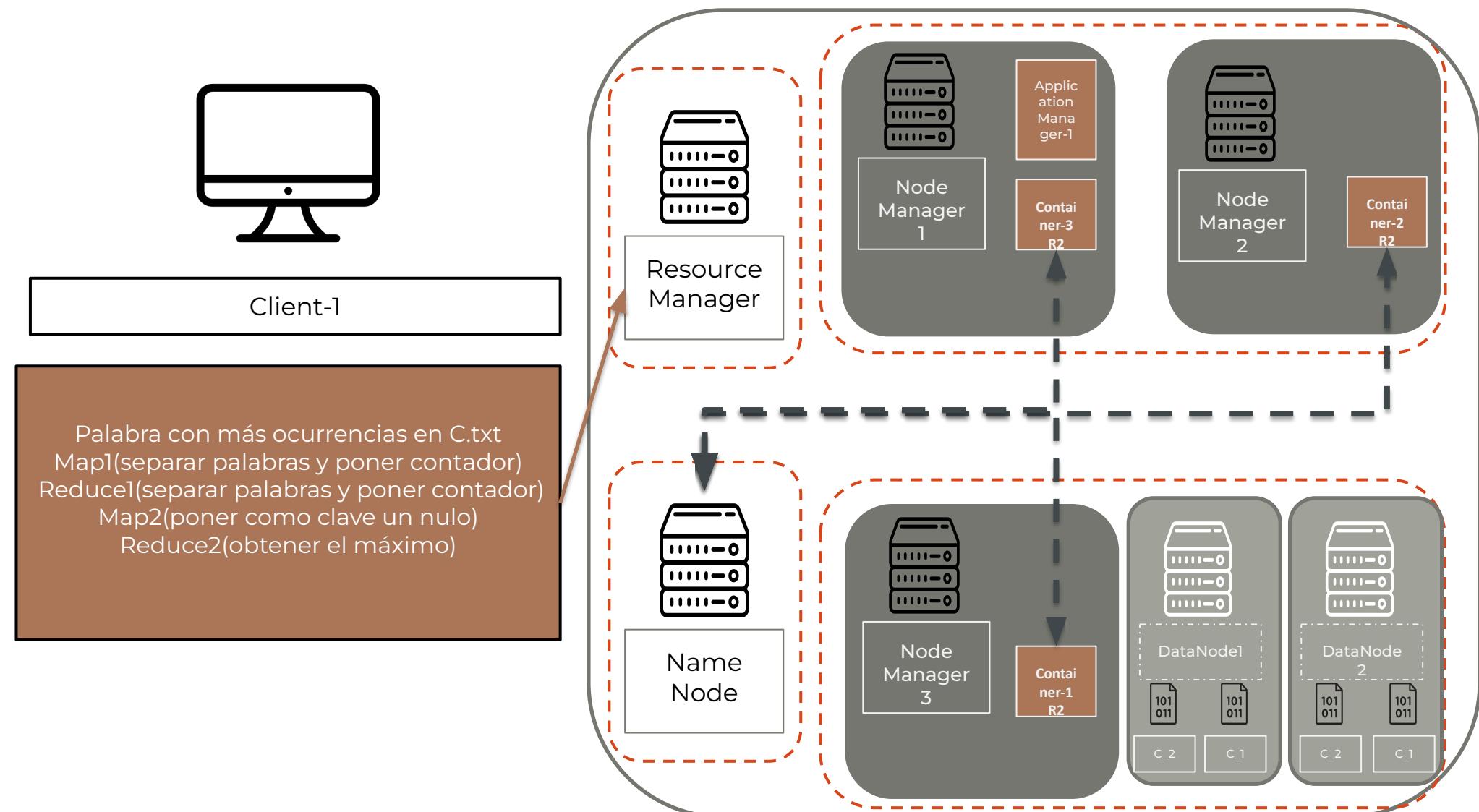
YARN - M&R - HDFS Integración



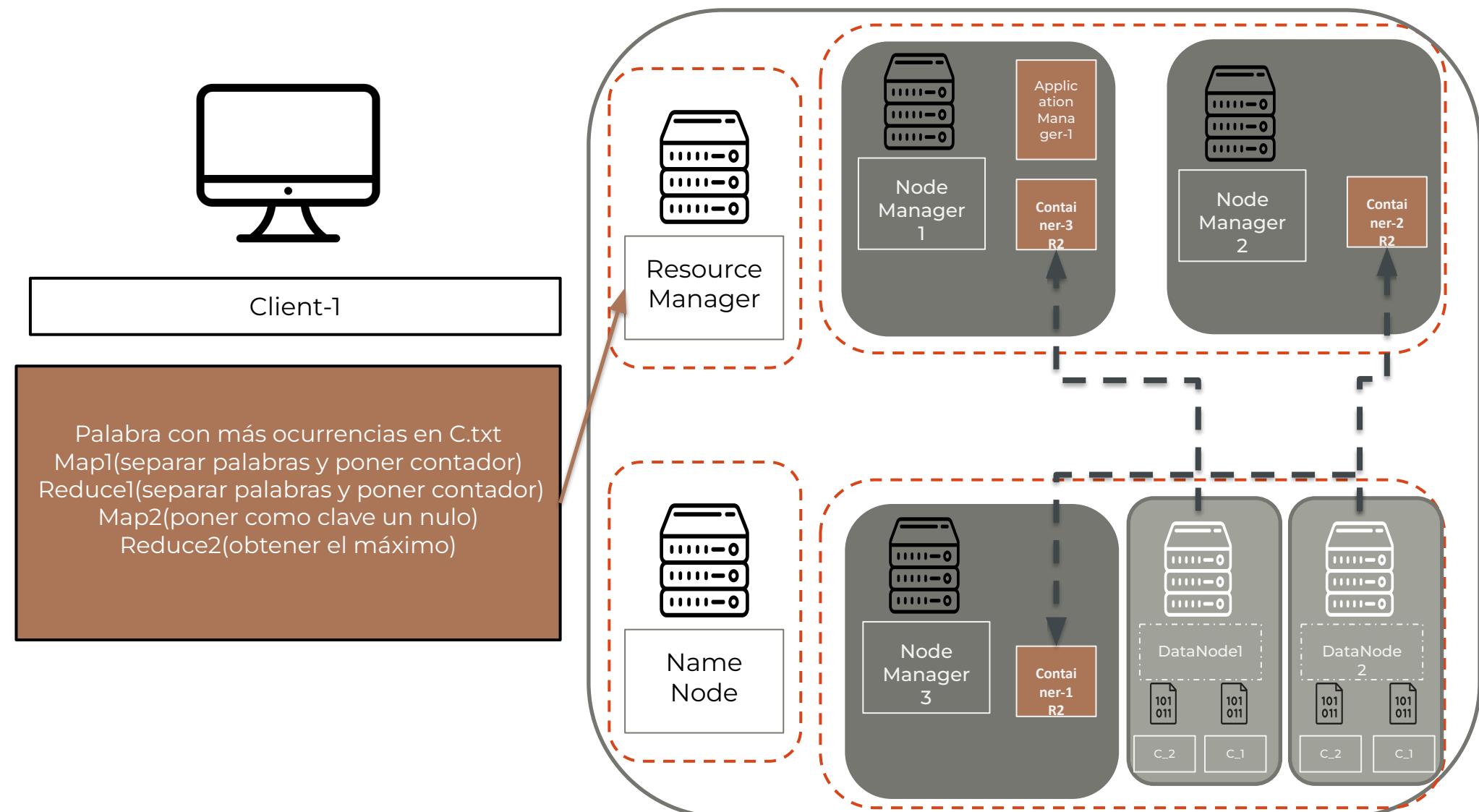
YARN - M&R - HDFS Integración



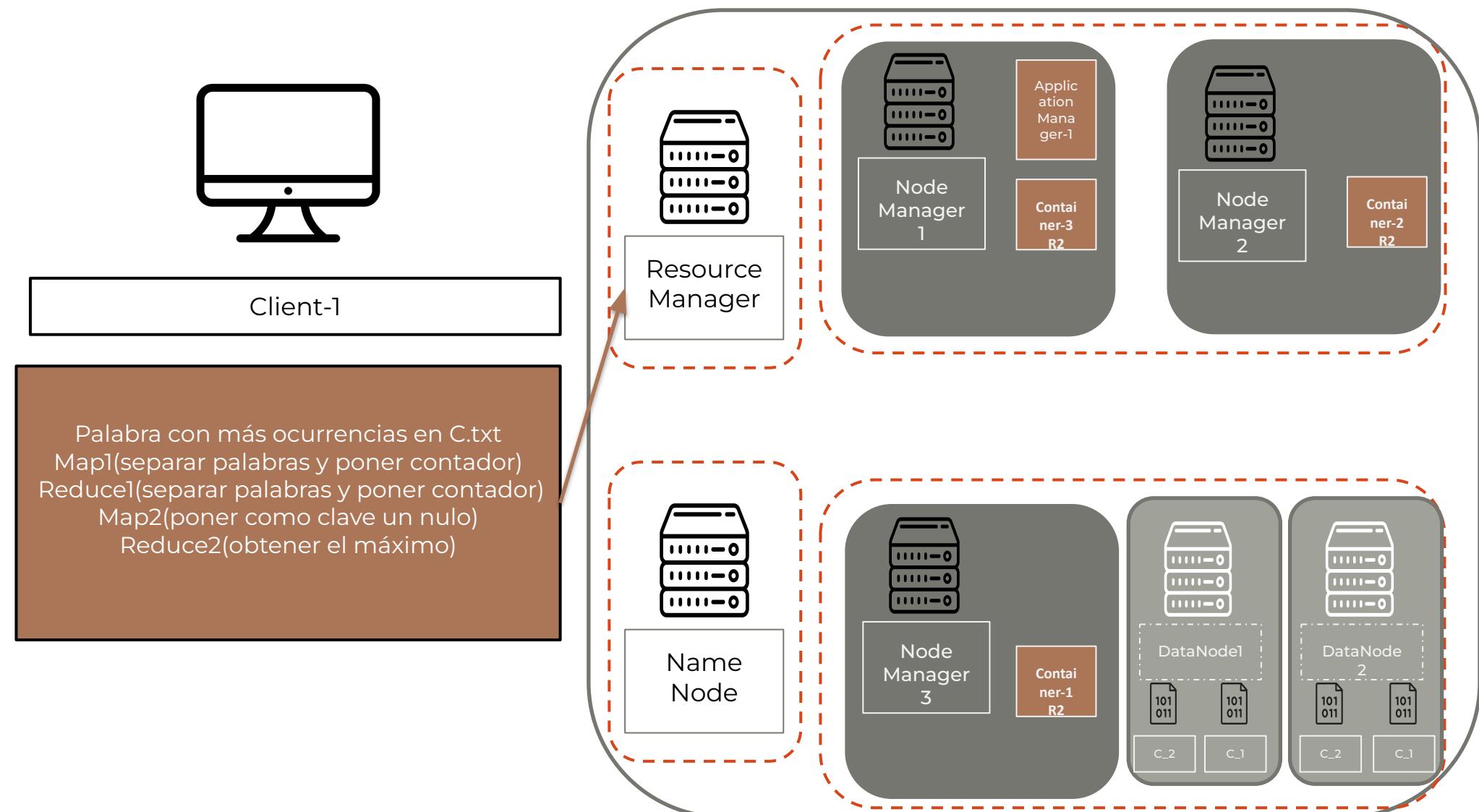
YARN - M&R - HDFS Integración



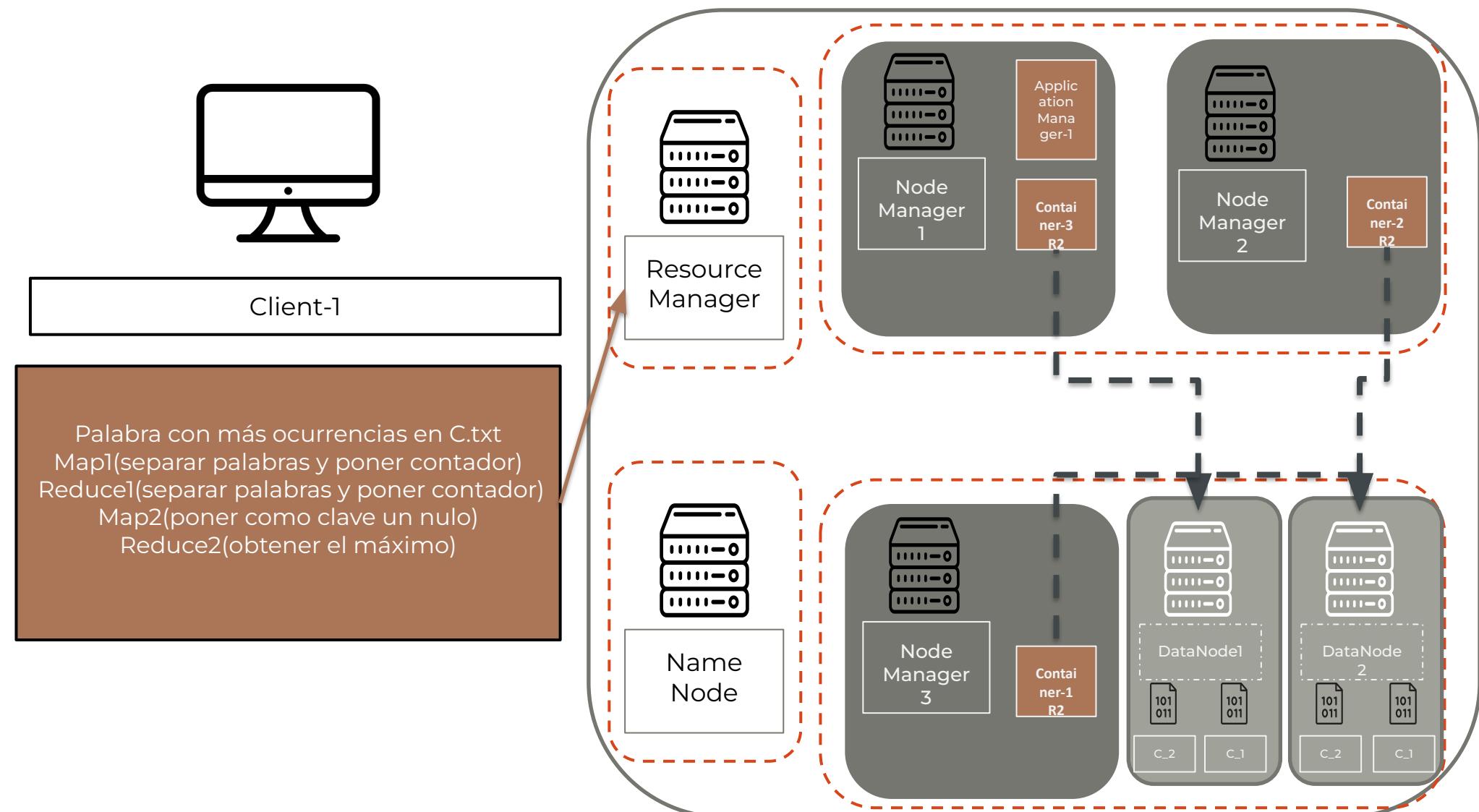
YARN - M&R - HDFS Integración



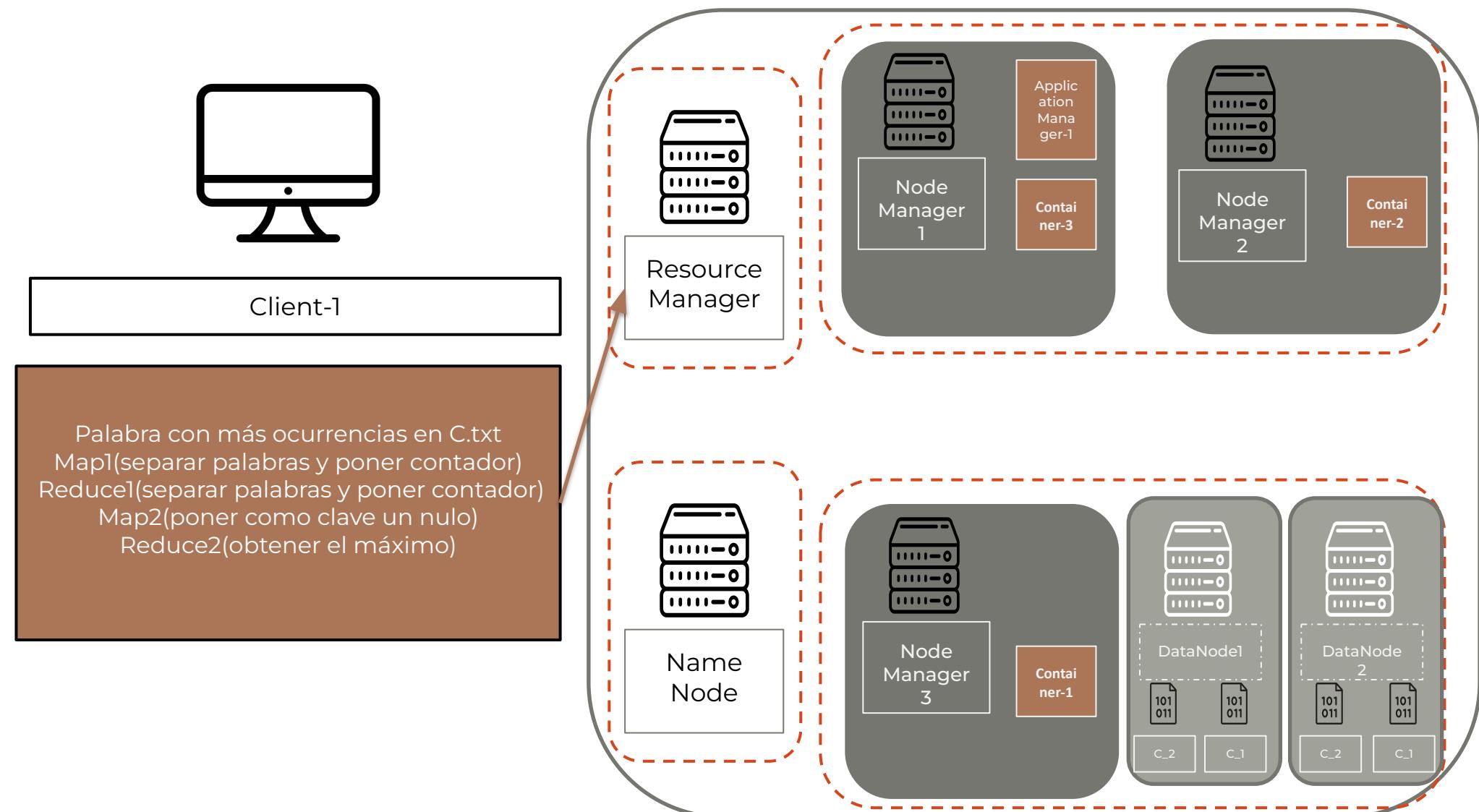
YARN - M&R - HDFS Integración



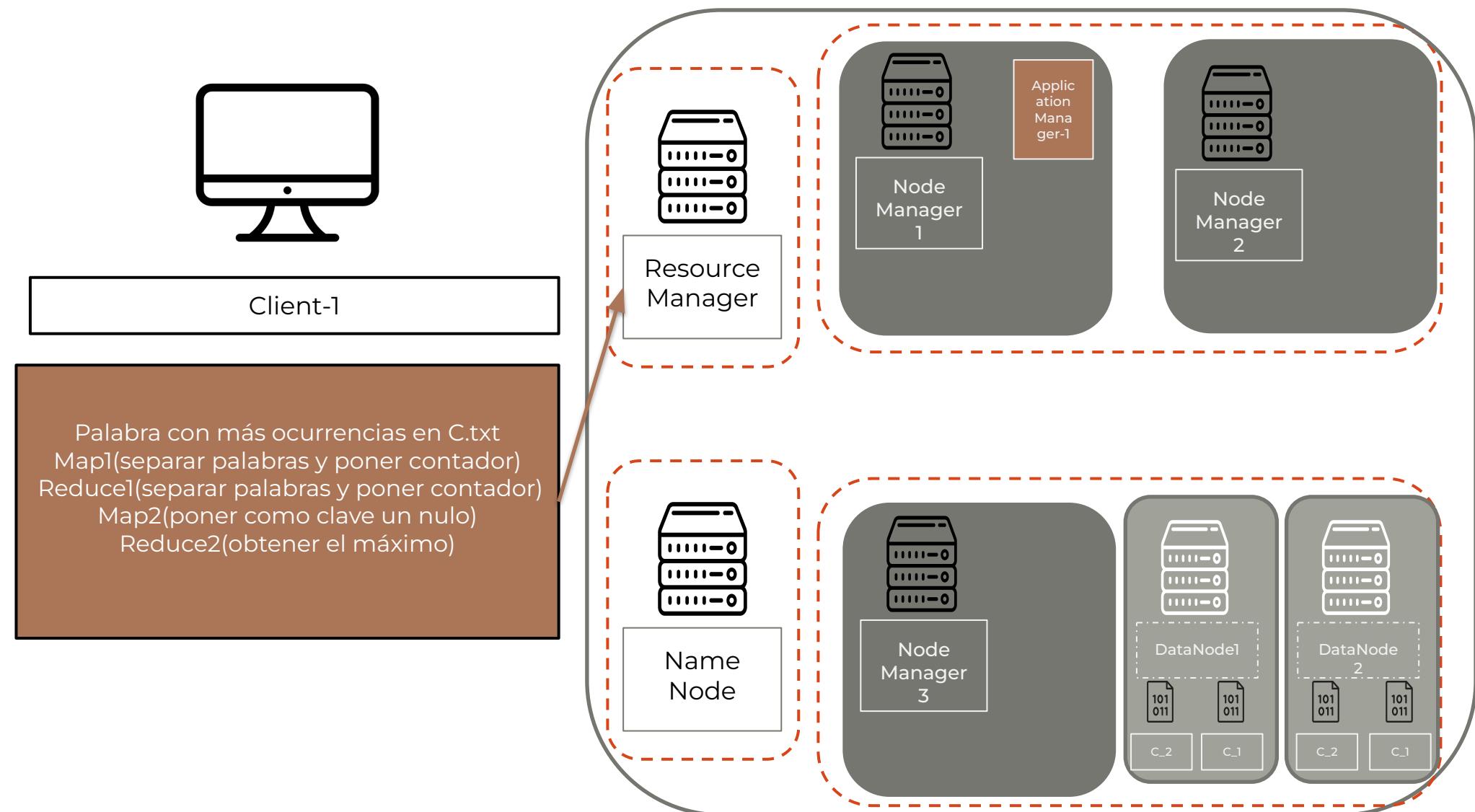
YARN - M&R - HDFS Integración



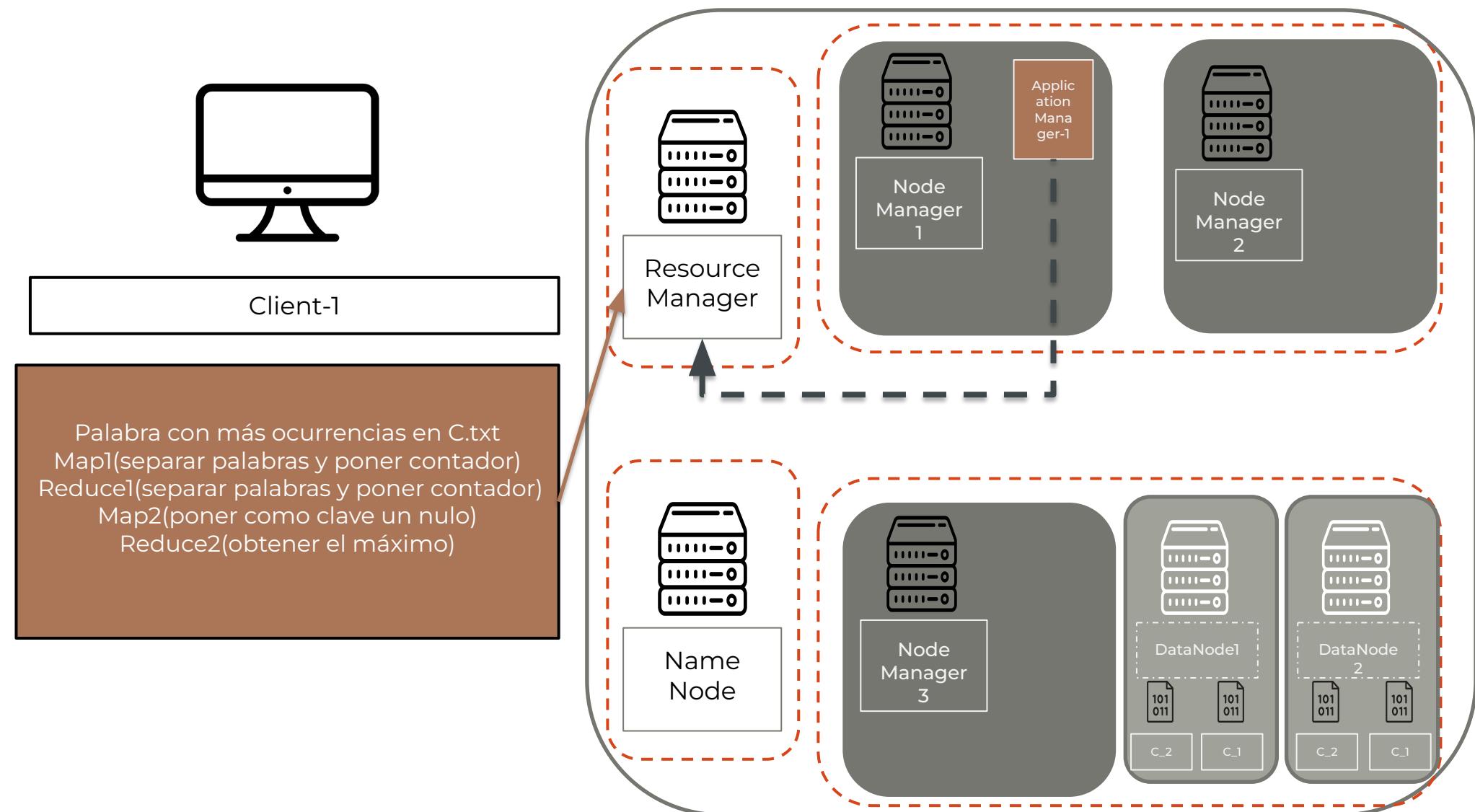
YARN - M&R - HDFS Integración



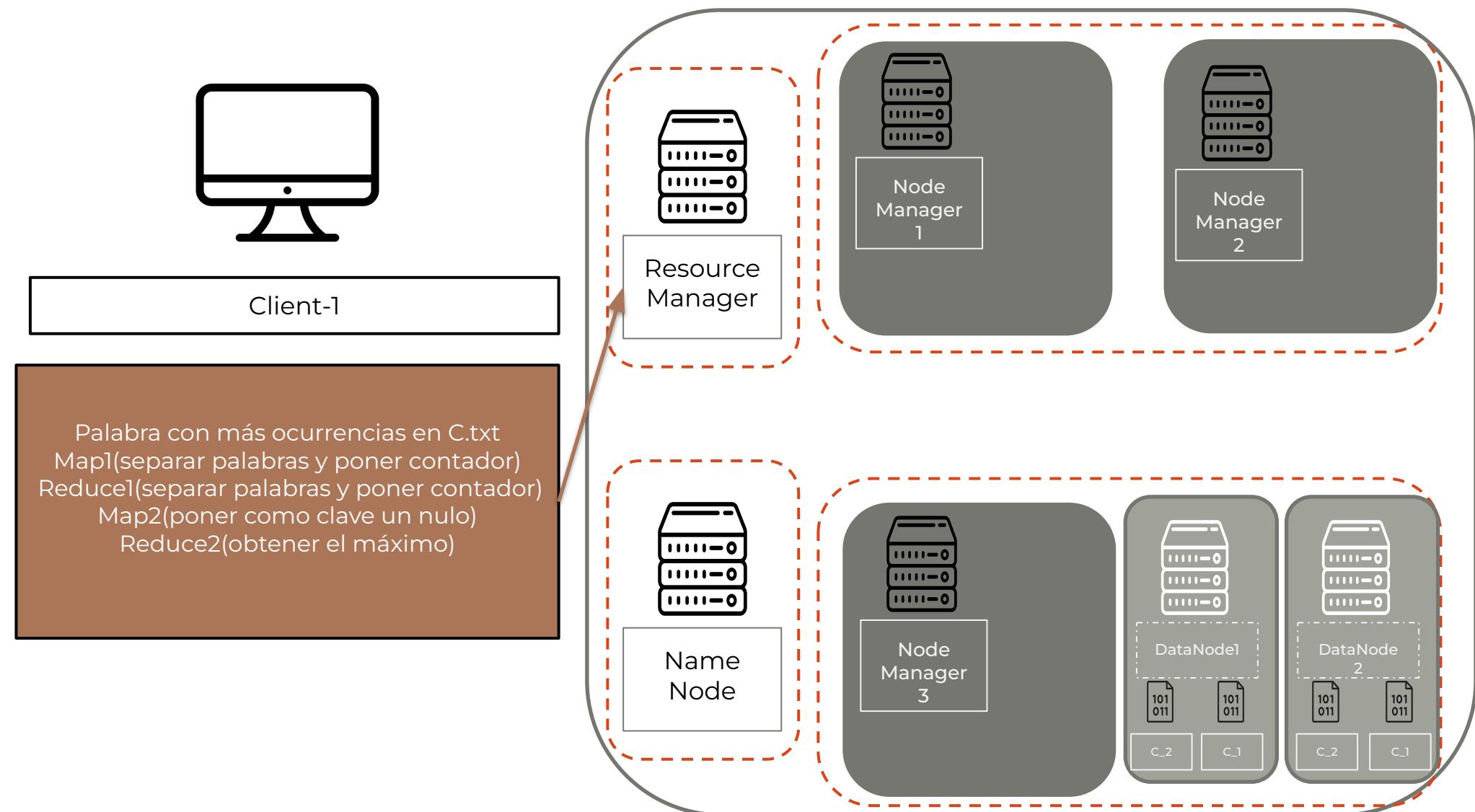
YARN - M&R - HDFS Integración



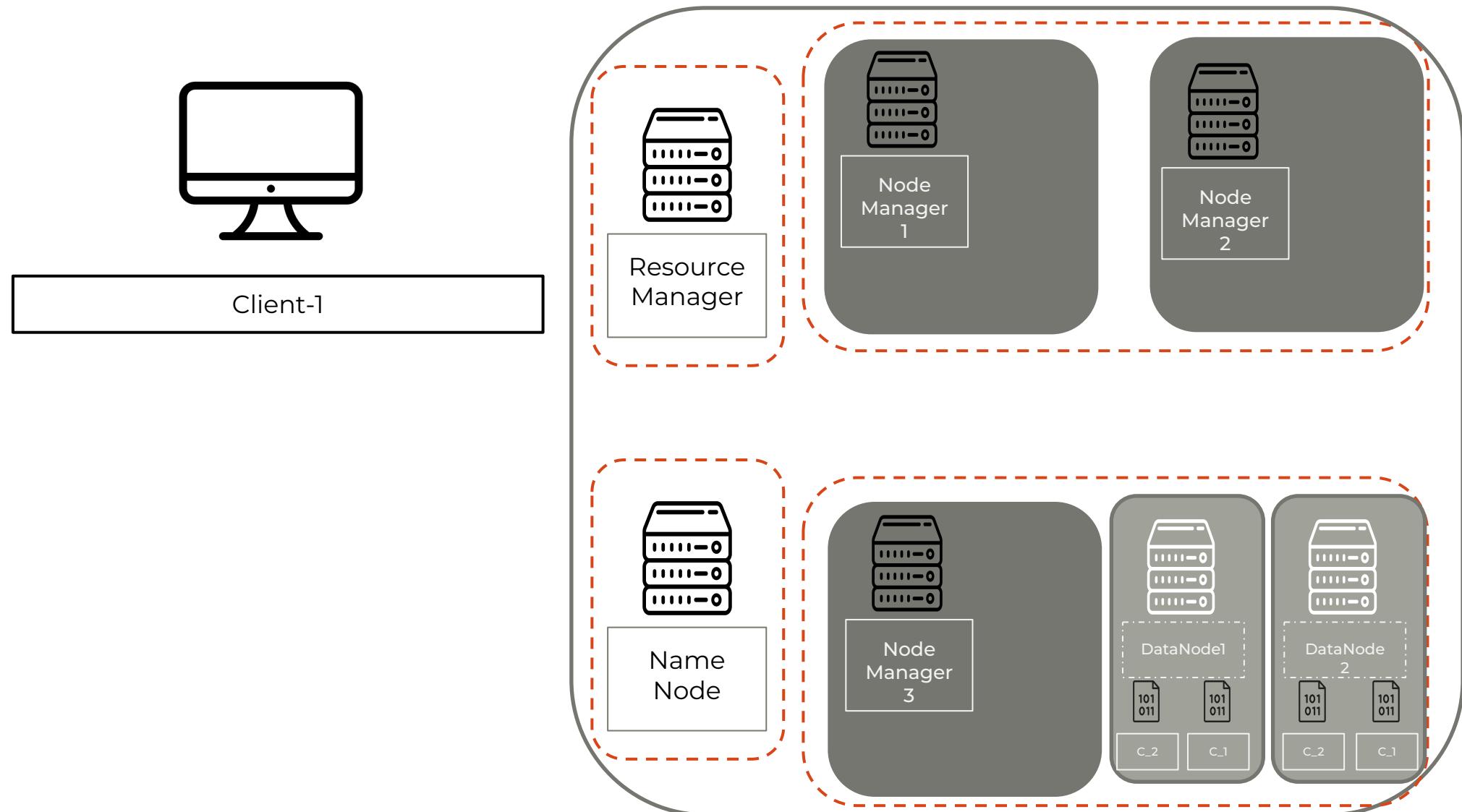
YARN - M&R - HDFS Integración



YARN - M&R - HDFS Integración

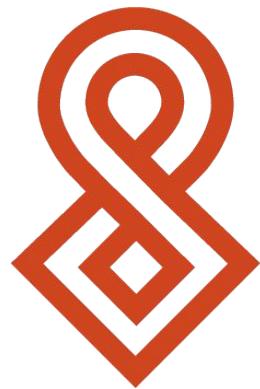


YARN - M&R - HDFS Integración



YARN- HA

- Una vez asignados los recursos, el cliente sólo se comunica con el **ResourceManager**
- **ResourceManager** es un único punto de fallo
- **YARN** sigue una estructura dos **ResourceManagers(activo/en espera)** para el HA
- YARN en HA necesita tecnologías externas
 - Zookeeper



Afi Escuela

© 2022 Afi Escuela. Todos los derechos reservados.