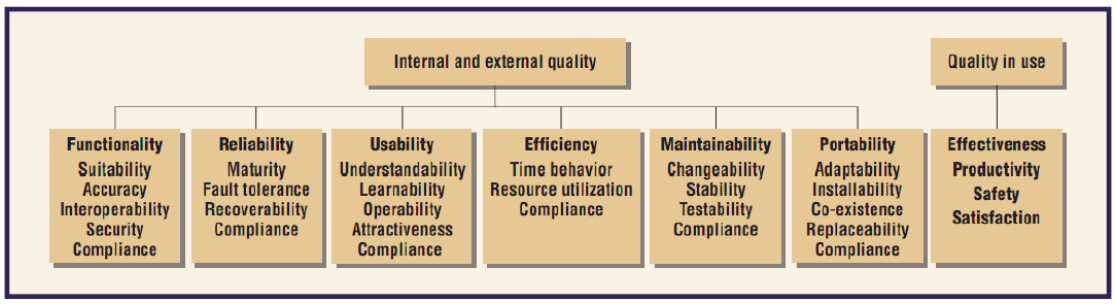
8 Architektury softwarových systémů, komponentové architektury, vzdálená invokace, distribuované komponentové architektury (CORBA), redundance a návrh spolehlivých systémů. (A4B77ASS)

8.1 Nefunkční požadavky softwarových systémů

Specifikace **ISO/IEC 9126**:

- **Functionality** – existence of a set of functions and their specified properties
- **Reliability** - capability of software to *maintain* its level of performance under stated conditions for a stated period of time
- **Usability** - effort needed for use
- **Efficiency** - relationship between the *level of performance* of the software and the amount of *resources* used, under stated conditions
- **Maintainability** - effort needed to *make specified modifications*
- **Portability** - transferred from one *environment* to another



8.2 Návrhové vzory pro distribuované systémy

- **Façade** - zakrývá komplexitu a heterogenitu systému či knihovny za jednoduché rozhraní, často zakrývá více objektů

- **Adapter/Wrapper** - poskytuje mapování mezi dvěma rozhraními se stejnou funkcionalitou, zaručuje kompatibilitu volání
- **Wrapper Façade** - zapouzdřuje funkce a data poskytované ne-objektově orientovaným API pod objektově orientované rozhraní
- **Proxy** - je lokální reprezentace vzdálených objektů, rozhraní nebo knihoven
- **Active Object** - odděluje spouštění metody od její invokace, tím zlepšuje souběžnost a zjednodušuje synchronní přístup k objektům
- **Reactor** - umožňuje událostmi řízeným aplikacím zpracovávání požadavků od jednoho či více klientů
- **Proactor** - umožňuje událostmi řízeným aplikacím zpracovávání požadavků, které jsou vyvolány dokončením asynchronních operací
- **HalfSync/HalfAsync** - odděluje synchronní a asynchronní zpracovávání v souběžných systémech

8.3 Vzdálená invokace - RMI

RMI (Remote Method Invocation) je způsob používání vzdálených objektů jako lokálních, dostupné jen pro Javu. Architektura klient-server: server vytvoří vzdálené objekty, klient tyto objekty získá a může na nich volat metody. Umožňuje načítat definice tříd za běhu.

Vzdálené objekty je možné předat jiné (klientské) VM pomocí reference, poté je na nich možné volat metody. Výpočet pak probíhá na VM serveru. Vzdálené objekty musí extendovat rozhraní Remote a všechny metody musí vyhazovat RemoteException. Rozhraní Task společné pro klienta i server definuje jejich vzájemnou komunikaci. Takto může server vykonávat jakékoliv úkoly, které implementují rozhraní Task, RMI pak za běhu načte definice tříd. Data přesouvané mezi klientem a servem musí být buď primitivní datové typy nebo objekty implementující Serializable nebo Remote.

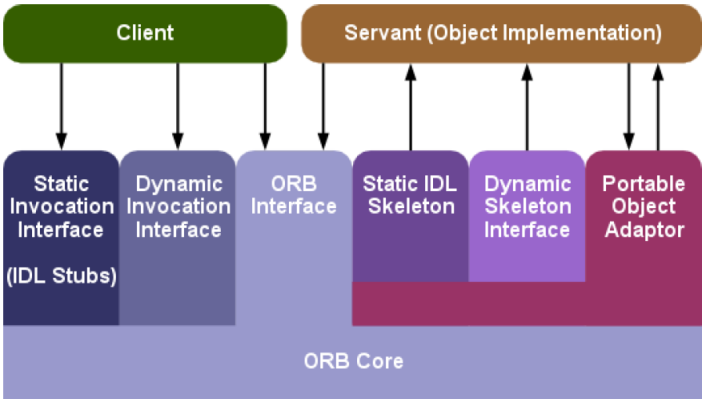
```
public interface Task<T> {
    T execute();
}

public interface Compute extends Remote {
    <T> T perform(Task<T> t) throws RemoteException;
}
```

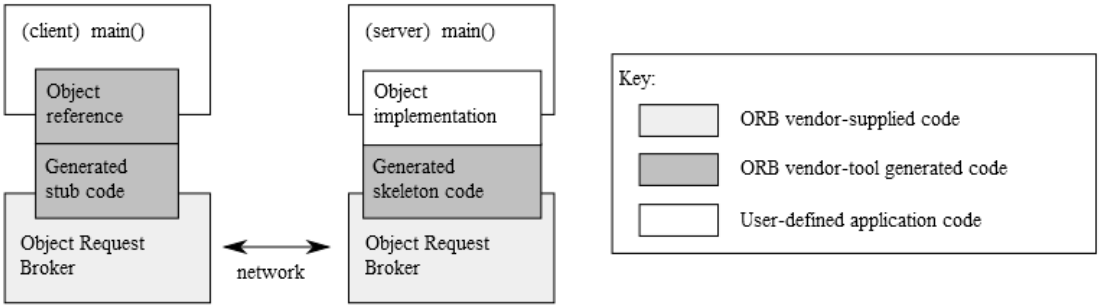
8.4 Distribuované komponenty - CORBA

CORBA (Common Object Request Broker Architecture) je platformně a jazykově nezávislý standard (architektura) pro distribuované výpočty. Umožňuje transparentní invokaci objektů a metod přes síť.

ORB (Object Request Broker) přenáší žádosti od klienta na server a volá metody na vzdálených objektech, potom co server žádost zpracuje, tak ORB přenese odpověď zpět ke klientovi. Klient pak volá metody na lokální proxy. Pro komunikaci mezi jednotlivými ORB a pro přenos dat se používá **IIOP** (Internet Inter-ORB Protocol).

IDL (Interface Definition Language) je platformně a jazykově nezávislý objektově orientovaný jazyk určený ke specifikaci business level služeb a objektů. Používá se pro popis dostupných lokálních a serverových metod.





8.5 Masivně distribuované architektury

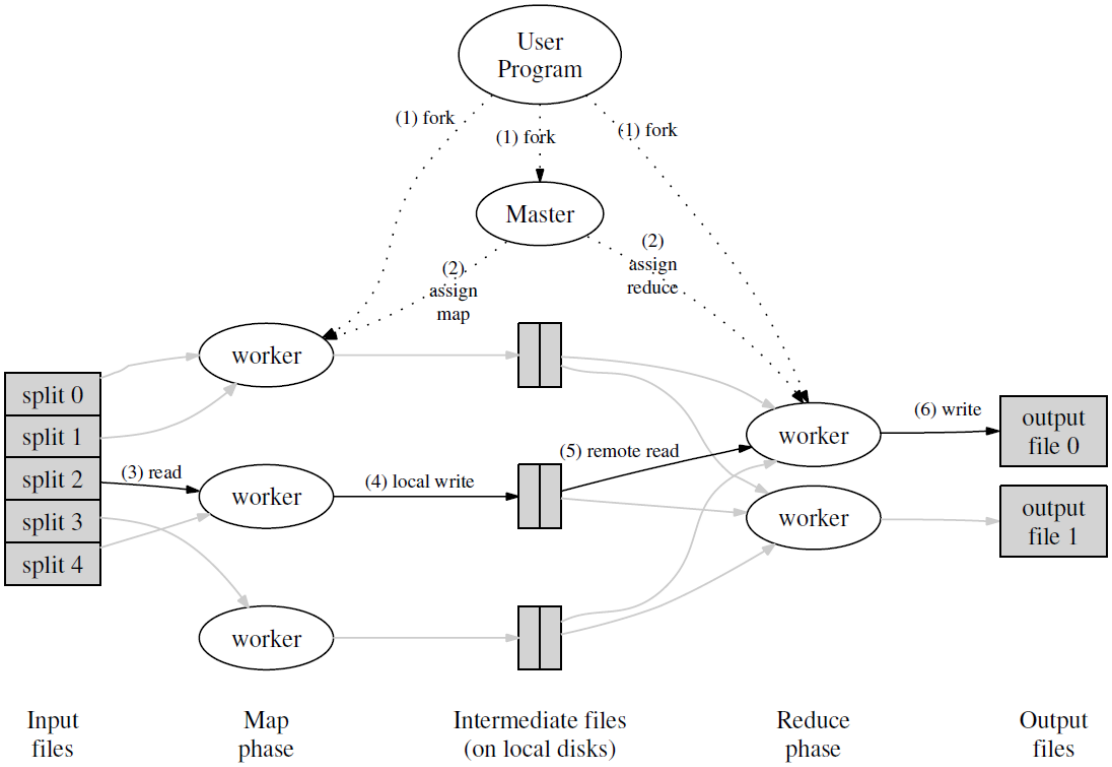
8.5.1 Map-Reduce

Navržen a používán Googlem, masivně paralelní přístup založený a inspirovaný na funkcionálním programování. Používá se k ukládání, manipulaci a hledání v datech s jednoduchou strukturou, kterých je ale velké množství. Operuje nad dvojicemi klíč-hodnota. Metoda Map-Reduce má dvě opakující se fáze:

- **1) Map** - funkce (filtr) se aplikuje na množinu záznamů/elementů v listu
- **2) Reduce** - zkrácení listu za použití nějaké agregační funkce

Z výpočetních jednotek je vybrán jeden řídící *master node* a ostatní jsou *worker nodes*. Masivní paralelizace je dosažena tím, že se data rozdělí do několika částí a nechají se

zpracovat různým *worker nodes*. Příklady použití: distribuované zpracování regulárních výrazu, počítání URL referencí, vytváření reverzního grafu webu, sémantické vyhledávání, invertovaní indexování, distribuované řazení.

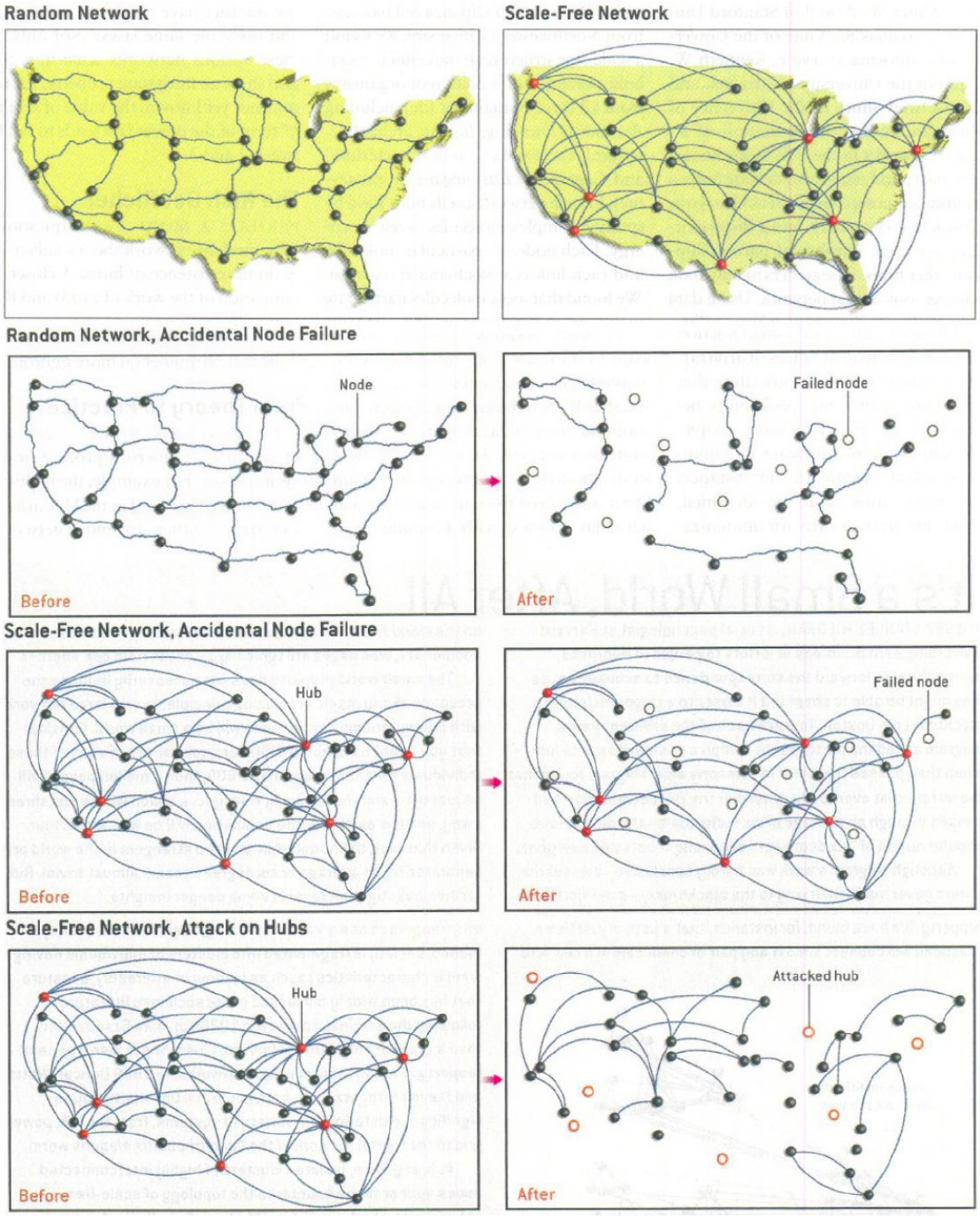


8.5.2 KaZaA

Je skupina protokolů a technologií používaných pro *peer-to-peer* komunikaci.

- **Napster** - *peer-to-peer* síť s centralizovaným řídícím prvkem
- **Gnutella** - Je *flat peer-to-peer* síť kde jsou si všichni peery rovni. Při startu se vytvoří daný počet náhodných spojení s peery, které jsou v tu dobu aktivní. Peery sdílí informace o ostatních peerech. V novějších verzích představen koncept *ultrapeerů* (peery s větším množstvím spojení, huby) - design ovlivněný *scale-free* sítěmi.

Scale-Free Networks Sítě reflektující povahu reálného světa. Existují dva typy nodů - *Ordinary Nodes* a *Super Nodes*. Při startu se ON připojí k některému z dostupných SN. Každý SN si udržuje databázi připojených ON a aktuální topologii dalších SN. Při hledání se ON spojí s SN, ale jsou možné spojení přímo ON k ON (Skype).


Random Network
Scale-Free Network
Random Network, Accidental Node Failure
Node
Before
Failed node
After
Scale-Free Network, Accidental Node Failure
Hub
Before
Failed node
After
Scale-Free Network, Attack on Hubs
Hub
Before
Attacked hub
After